TURBO VISION

для языка Паскаль

Справочник

ФИРМА „ДИАЛЕКТИКА"

НОВЫЕ ТЕХНОЛОГИИ В ОБЛАСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

TURBO VISION

для языка Паскаль

Справочник

«Диалектика»
Киев • 1992

ББК 32.973
Т 43

Слайд артфотостудии "AFFA"

Т 43 Turbo Vision для языка Pascal. Справочник.
М: «И.В.К.–Софт», 1992 — 288с.
ISBN 5–85477–001–6

Книга содержит полную справочную информацию о всех стандартных объектах и методах объектно-ориентированной библиотеки Turbo Vision. Справочник служит дополнением к первой части Описания.

Для программистов разной квалификации, для всех использующих язык программирования Turbo Pascal.

ББК 32.973

ISBN 5–85477–001–6

КАК ИСПОЛЬЗОВАТЬ СПРАВОЧНИК

Справочник по Turbo Vision описывает все стандартные объекты и методы в иерархии Turbo Vision вместе с мнемоническими идентификаторами, константами и записями, необходимыми для разработки программ на Turbo Vision. Справочник не задумывался как учебник.

По своей природе сложные библиотеки объектов, такие, как в Turbo Vision, имеют множество компонент. Для того, чтобы избежать бесконечного повторения материала, мы поместили наиболее полную информацию в алфавитных разделах (глава 3 и 4) вместе с другими менее детальными материалами, которые позволяют вам просматривать компоненты Turbo Vision в их иерархических и физических взаимосвязях с ссылками на более детальную информацию.

КАК НАЙТИ ИНФОРМАЦИЮ

Глава 2 «Справочник по модулям» описывает модули Turbo Vision. Она включает список всех типов, констант, переменных, процедур и функций, объявленных в каждом модуле.

Глава 4 «Глобальные ссылки» приводит все глобальные константы, переменные, процедуры и функции Turbo Vision, т.е. если то, что вы ищете — не объект и не часть объекта, вы найдете его здесь.

Глава 3 «Справочник по объектам» приводит в алфавитном порядке все стандартные типы объектов Turbo Vision, включая все их поля и методы.

Запомните, что эта глава описывает только те аспекты каждого объекта, которые принадлежат ему. Большинство объектов имеют поля и методы, наследуемые от других

объектов. Так, если вы хотите найти метод для объекта, вначале проверьте этот объект. Если вы не найдете метод в этом объекте, проверьте его непосредственного предка. Диаграмма в начале описания каждого объекта объясняет его взаимосвязи с предками и непосредственными потомками.

ОБЪЕКТЫ ВООБЩЕ

Вспомним, что каждый объект (кроме базового объекта TObject и двух специальных объектов TPoint и TRect) наследует поля и методы родительского объекта. Порожденные вами объекты будут также наследовать поля и методы предка. Многие стандартные объекты имеют абстрактные методы, которые должны быть перекрыты в порожденном объекте. Другие методы виртуальные, это означает, что обычно вам необходимо перекрыть их. Существуют также методы, которые выполняют полезные действия, если они не были перекрыты.

СОГЛАШЕНИЯ ОБ ИМЕНОВАНИИ

Все стандартные типы объектов в Turbo Vision имеют набор имен, использующих мнемонические префиксы. Первая буква идентификатора говорит вам, используете ли вы тип объекта, указатель на него, его регистрационную запись в потоке или его палитру цветов.

- Тип объекта начинается с T: TObject.
- Указатели на объекты начинаются с P:
  PObject = ^TObject.
- Регистрационные записи потоков начинаются с R:
  RObject.
- Палитры цветов начинаются с C: CObject.

Все константы Turbo Vision имеют двухсимвольные мнемонические префиксы, указывающие их использование.

Таблица 1.1

Префиксы констант Turbo Vision

Префикс	Назначение	Пример
ap	Палитра программы	apColor
bf	Флаг кнопки	bfNormal
cm	Команда	cmQuit
co	Код коллекции	coOverFlow
dm	Режим перемещения	dmDragGrow
ev	Константа события	evMouseDown
gf	Флаг режима перемещения	gfGrowLoX
hc	Контекст помощи	hcNoContent
kb	Константа клавиатуры	kbAltX
mb	Кнопка мышки	mbLeftButton
of	Флаг опций	ofTopSelect
sb	Полоса скроллинга	sbLeftArrow
sf	Флаг состояния	sfVisible
sm	Режим экрана	smMono
st	Код потока	stOK
wf	Флаг окна	wfMove
wn	Номер окна	wnNoNumber
wp	Палитра окна	wpBlueWindow

СПРАВОЧНИК ПО МОДУЛЯМ

Эта глава кратко описывает содержимое каждого из модулей Turbo Vision. Вначале мы дадим обзор модулей Turbo Vision, а затем более детально опишем каждый модуль.

Turbo Vision содержит 9 модулей:

Таблица 2.1

Модули Turbo Vision

Модуль	Содержимое
App	Все определения объектов для написания программ, управляемых от событий
Dialogs	Инструменты и элементы управления
Drivers	Поддержка мышки
HistList	Списки историй для строк ввода
Memory	Система управления памятью
Menus	Объекты для добавления меню и строки статуса
Objects	Основные определения объектов, включая все типы объектов для потоков, коллекций и ресурсов
TextView	Видимые элементы для представления текста
Views	Основные объекты для использования окон в программе: видимые элементы, окна, рамки, полосы скроллинга и т.д.

МОДУЛЬ OBJECTS

Модуль Objects содержит основные определения объектов Turbo Vision, включая базовый объект иерархии Turbo Vision TObject, а также все невидимые элементы Turbo Vision: потоки, коллекции и ресурсы.

ТИПЫ

Типы преобразования записей

Тип	Использование
FNameStr	Строка имени файла DOS
LongRec	Преобразует Longint в старшее и младшее слово
PChar	Указатель для динамического распределения символа
PString	Указатель для динамических строк
PtrRec	Преобразует Pointer в сегмент и смещение
TByteArray	Массив значений Byte, используемый для приведения типов
TWordArray	Массив значений Word, используемый для приведения типов
WordRec	Преобразует Word в старший и младший байт

Типы модуля Objects

Тип	Использование
TBufStream	Буферизованный поток DOS Turbo Vision
TCollection	По существу полиморфный массив

Тип	Использование
TDosStream	Поток Turbo Vision для файла DOS
TEmsStream	Поток Turbo Vision в EMS памяти
TItemList	Массив указателей, используемый коллекциями
TObject	Базовый объект иерархии Turbo Vision
TPoint	Объект для построения точки на экране
TRect	Объект из двух точек для определения области на экране
TResourceCollection	Специализированный TCollection для ресурсов
TResourceFile	Объект для сохранения ресурсов на диске
TSortedCollection	Специализированный TCollection для автоматической сортировки
TStream	Базовый тип определения потока Turbo Vision
TStreamRec	Запись регистрации потока
TStrIndex	Массив TStrIndexRec
TStrIndexRec	Запись строковых индексов, используемая TStrIndex
TStringCollection	Специализированный TSortedCollection для строк
TStringList	Объект списка, используемый для ресурсов строк
TStrListMaker	Специальный объект для создания списков строк

Режимы доступа к потоку

Константа	Значение	Назначение
stCreate	$3C00	Создает новый файл
stOpenRead	$3D00	Доступ только для чтения
stOpenWrite	$3D01	Доступ только для записи
StOpen	$3D02	Доступ для чтения и записи

Коды ошибок потока

Код ошибки	Значение	Назначение
stOk	0	Нет ошибки
stError	−1	Ошибка доступа
stInitError	−2	Не может инициализировать поток
stReadError	−3	Чтение за концом потока
stWriteError	−4	Не может расширить поток
stGetError	−5	Чтение незарегистрированного типа объекта
stPutError	−6	Запись незарегистрированного типа объекта

Максимальный размер коллекции

Константа	Значение	Назначение
MaxCollectionSize	16380	Максимальный размер TCollection

Коды ошибок коллекции

Код ошибки	Значение	Назначение
coIndexError	−1	Индекс вне диапазона
coOverflow	−2	Переполнение

ПЕРЕМЕННЫЕ

Переменная	Тип	Начальное значение	Назначение
EmsCurHandle	Word	$FFFF	Текущий обработчик EMS
EmsCurPage	Word	$FFFF	Текущая страница EMS

ПРОЦЕДУРЫ И ФУНКЦИИ

Процедура	Операция
Abstract	Процедура по умолчанию для методов, которые должны быть перекрыты
DisposeStr	Удаляет строку, созданную с помощью NewStr
RegisterType	Регистрирует тип объекта в потоках Turbo Vision

Функция	Операция
LongDiv	Деление длинного целого на целое
LongMul	Умножение двух целых в длинное целое
NewStr	Распределение строки в куче

МОДУЛЬ VIEWS

Модуль Views содержит основные компоненты видимых элементов. Это оба абстрактных типа, таких, как TView и TGroup, и полезные компоненты более сложных групп, таких, как рамки окон и полосы скроллинга. Более сложные видимые элементы находятся в модулях Dialogs и TextView.

ТИПЫ

Тип	Использование
TCommandSet	Разрешает и запрещает группы команд
TDrawBuffer	Буфер, используемый для методов отрисовки
TFrame	Рамка объекта, используемая окнами
TGroup	Абстрактный объект для сложных видимых элементов
TListViewer	Базовый тип для окон списков и т.п.
TPalette	Тип палитры, используемой всеми видимыми элементами
TScrollBar	Объект определяющий полосу скроллинга
TScrollChars	Символьные компоненты полосы скроллинга
TScroller	Базовый объект для скроллинга текста в окнах
TTitleStr	Строка заголовка, используемая TFrame
TVideoBuf	Видеобуфер, используемый монитором экрана
TView	Абстрактный объект; основа всех видимых объектов

Тип	Использование
TWindow	Базовый объект для окон изменяющих размеры

КОНСТАНТЫ

Маски State для TView

Константа	Значение	Назначение
sfVisible	$0001	Видимый элемент виден
sfCursorVis	$0002	Видимый элемент имеет видимый курсор
sfCursorIns	$0004	Курсор видимого элемента — блок для режима вставки
sfShadow	$0008	Видимый элемент имеет тень
sfActive	$0010	Видимый элемент или его владелец — активное окно
sfSelected	$0020	Видимый элемент — владелец выбранного видимого элемента
sfFocused	$0040	Видимый элемент активен
sfDragging	$0080	Видимый элемент — перемещаемый
sfDisabled	$0100	Видимый элемент запрещен
sfModal	$0200	Видимый элемент в модальном состоянии
sfExposed	$0800	Видимый элемент присоединен к программе

Константы модуля Views

Константа	Значение	Назначение
hcNoContext	0	Неопределенный код контекста подсказки
hcDragging	1	Контекстная подсказка пока объект перемещается
MaxViewWidth	132	Максимальная длина видимого элемента в символах
wnNoNumber	0	Номер TWindow

Маски Option для TView

Константа	Значение	Назначение
ofSelectable	$0001	Видимый элемент может быть выбран
ofTopSelect	$0002	Выбираемый видимый элемент перемещается на вершину владельца
ofFirstClick	$0004	Отметка мышкой выбирает и производит действие
ofFramed	$0008	Видимый элемент имеет видимую рамку
ofPreProcess	$0010	Видимый элемент встретил активные события раньше активного видимого элемента
ofPostProcess	$0020	Видимый элемент встретил активные события позже активного видимого элемента
ofBuffered	$0040	Группа может иметь кэш-буфер

Константа	Значение	Назначение
ofTileable	$0080	Видимый элемент может располагаться черепицей на панели экрана
ofCenterX	$0100	Центр видимого элемента расположен горизонтально внутри владельца
ofCenterY	$0200	Центр видимого элемента расположен вертикально внутри владельца
ofCentered	$0300	Центр видимого элемента расположен горизонтально и вертикально внутри владельца

Маски GrowMode для TView

Константа	Значение	Назначение
gfGrowLoX	$01	Левая сторона соответствует правой стороне владельца
gfGrowLoY	$02	Верх соответствует низу владельца
gfGrowHiX	$04	Правая сторона соответствует правой стороне владельца
gfGrowHiY	$08	Низ соответствует низу владельца
gfGrowAll	$0F	Видимый элемент следует нижнему правому углу владельца
gfGrowRel	$10	Сохраняет относительный размер, когда изменяется размер экрана

Маски DragMode для TView

Константа	Значение	Назначение
dmDragMove	$01	Видимый элемент может перемещаться
dmDragGrow	$02	Видимый элемент может изменять размер
dmLimitLoX	$10	Левая сторона видимого элемента не может выходить за Limits
dmLimitLoY	$20	Верх видимого элемента не может выходить за Limits
dmLimitHiX	$40	Правая сторона видимого элемента не может выходить за Limits
dmLimitHiY	$80	Низ видимого элемента не может выходить за Limits
dmLimitAll	$F0	Ни одна часть видимого элемента не может выходить за Limits

Коды полосы скроллинга

Константа	Значение	Назначение
sbLeftArrow	0	Левая горизонтальная стрелка полосы
sbRightArrow	1	Правая горизонтальная стрелка полосы
sbPageLeft	2	Левая горизонтальная страничная область полосы

Константа	Значение	Назначение
sbPageRight	3	Правая горизонтальная страничная область полосы
sbUpArrow	4	Вертикальная стрелка вверх полосы
sbDownArrow	5	Вертикальная стрелка вниз полосы
sbPageUp	6	Вертикальное направление вверх страничной области полосы
sbPageDown	7	Вертикальное направление вверх страничной области полосы
sbIndicator	8	Индикатор полосы скроллинга

Маски флага окна

Константа	Значение	Назначение
wfMove	$01	Верхняя строка рамки может перемещать окно
wfGrow	$02	Рамка окна имеет угол изменения размера
wfClose	$04	Рамка окна имеет закрывающую кнопку
wfZoom	$08	Рамка окна имеет кнопку масштабирования

Элементы палитры TWindow

Константа	Значение	Назначение
wpBlueWindow	0	Текст в окне желтый на синем
wpCyanWindow	1	Текст в окне синий на бирюзовом
wpGrayWindow	2	Текст в окне черный на сером

Стандартные команды видимого элемента

Константа	Значение	Назначение
cmReceivedFocus	50	Видимый элемент получает активность
cmReleasedFocus	51	Видимый элемент освобождает активность
cmCommandSet-Changed	52	Множество команд изменилось
cmScrollBarChanged	53	Полоса скроллинга изменила значение
cmScrollBarClicked	54	Полоса скроллинга была отмечена
cmSelectWindowNum	55	Пользователь хочет выбирает окно по номеру
cmRecordHistory	56	Список истории может сохранять содержимое строки ввода

ПЕРЕМЕННЫЕ

Переменная	Тип	Начальное значение	Назначение
MinMinSize	TPoint	(X: 16; Y: 6)	Минимальный размер окна
ShadowSize	TPoint	(X: 2; Y: 1)	Размер тени окна
ShadowAttr	Byte	$08	Атрибут окна

ФУНКЦИИ

Функция	Операция
Message	Пересылает сообщения

МОДУЛЬ DIALOGS

Модуль Dialogs определяет большинство элементов наиболее часто используемых при создании диалоговых окон. Это включает сами диалоговые окна (которые являются специализированными окнами) и различные элементы управления, такие как кнопки, метки, зависимые и независимые кнопки, строки ввода и списки истории.

ТИПЫ

Тип	Использование
TButton	Нажатия кнопок для генерации команд
TCheckBoxes	Кластеры с включением и выключением кнопок
TCluster	Абстрактный тип для зависимых и независимых кнопок
TDialog	Специализированное окно для диалоговых окон

Тип	Использование
THistory	Список предыдущих элементов для строки ввода
TInputLine	Редактор текстового ввода
TLabel	Метка для кластера или строки ввода
TListBox	Скроллингуемый список для выбора пользователем
TParamText	Форматированный статический текст
TRadioButtons	Кластер или кнопки, только одна из которых может быть нажата
TSItem	Элементы строк в связанном списке, используемые кластерами
TStaticText	Простой текст

КОНСТАНТЫ

Флаги кнопок

Константа	Значение	Назначение
bfNormal	$00	Обычная кнопка
bfDefault	$01	Кнопка по умолчанию
bfLeftJust	$02	Кнопка текста может быть выровнена влево

ПРОЦЕДУРЫ И ФУНКЦИИ

Функция	Операция
NewSItem	Создает новый элемент строки для окна списка
Процедура	**Операция**
RegistersDialogs	Регистрирует все объекты в модуле Dialogs для использования с потоками

МОДУЛЬ APP

Модуль App (предоставлен в исходных кодах) обеспечивает элементы оболочки Turbo Vision. В APP определены четыре очень мощных объектных типа, включая объекты TApplication и TProgram, которые служат в качестве программ Turbo Vision и объект панели экрана, который управляет большинством элементов в оконных программах.

ТИПЫ

Тип	Использование
TApplication	Объект-программа с монитором событий, монитором экрана, обработкой ошибок и управлением памятью
TBackGround	Цвет фона для панели экрана
TDeskTop	Групповой объект для окон и диалоговых окон
TProgram	Абстрактный объект-программа

ПЕРЕМЕННЫЕ

Переменная	Тип	Начальное значение	Назначение
Application	PProgram	nil	Указатель на текущую программу
DeskTop	PDeskTop	nil	Указатель на текущую панель экрана
StatusLine	PStatusLine	nil	Указатель на текущую строку статуса
MenuBar	PMenuView	nil	Указатель на текущую полосу меню

МОДУЛЬ MENUS

Модуль Menus обеспечивает все объекты и процедуры для системы меню Turbo Vision, включая выпадающие меню и активные элементы строки статуса.

ТИПЫ

Тип	Использование
TMenu	Связанный список записей TMenuItem
TMenuBar	Связанный с меню горизонтальный заголовок
TMenuBox	Выпадающие окна меню
TMenuItem	Запись, связывающая метку текста, горячую клавишу, команду и контекстную подсказку
TMenuStr	Строковый тип для меток меню
TMenuView	Абстрактный объектный тип для полосы и окон меню
TStatusDef	Запись, связывающая контекстных подсказки со списком элементов строки статуса
TStatusItem	Строка сообщения внизу экрана программы, включающая список записей TStatusDef
TStatusLine	

ПРОЦЕДУРЫ И ФУНКЦИИ

Функции TMenuItem

Функция	Операция
NewItem	Создает новый элемент меню
NewLine	Создает строку окна меню
NewSubMenu	Создает подменю полосы меню или окна меню

Процедуры TMenu

Процедура	Операция
NewMenu function	Распределяет меню в куче
DisposeMenu procedure	Удаляет меню из кучи

Функции TStatusLine

Функция	Операция
NewStatusDef	Определяет диапазон контекстных подсказок и указатель на список элементов статуса
NewStatusKey	Определяет элемент строки статуса и связывает его с командой и горячей клавишей

МОДУЛЬ DRIVERS

Модуль Drivers содержит все специализированные драйверы Turbo Vision, включая драйверы мышки и клавиатуры, поддержку экрана и систему обработки ошибок с монитором событий для программ, управляемых событиями.

ТИПЫ

Тип	Использование
TEvent	Тип записи события
TSysErrorFunc	Функциональный тип обработчика системных ошибок

Маски состояния кнопок мышки

Константа	Значение	Назначение
mbLeftButton	$01	Левая кнопка мышки
mbRightButton	$02	Правая кнопка мышки

Коды событий

Константа	Значение	Назначение
evMouseDown	$0001	Кнопка мышки нажата
evMouseUp	$0002	Кнопка мышки освобождена
evMouseMove	$0004	Мышка изменила положение
evMouseAuto	$0008	Автоматический повтор события от мышки
evKeyDown	$0010	Событие — нажатие клавиши
evCommand	$0100	Событие — команда
evBroadcast	$0200	Событие — общее сообщение

Маски событий

Константа	Значение	Назначение
evNothing	$0000	Событие очищено
evKeyboard	$0010	Событие пришло от клавиатуры
evMouse	$000F	Событие пришло от мышки
evMessage	$FF00	Событие — сообщение или команда

Маски клавиатуры

Константа	Значение	Назначение,
kbRightShift	$0001	Нажат правый Shift
kbLeftShift	$0002	Нажат левый Shift
kbCtrlShift	$0004	Нажат Ctrl и Shift
kbAltShift	$0008	Нажат Alt и Shift
kbScrollState	$0010	Установлен Scroll lock
kbNumState	$0020	Установлен Num lock
kbCapsState	$0040	Установлен Caps lock
kbInsState	$0080	Включен режим Insert

Коды стандартных команд

Команда	Значение	Назначение
cmValid	0	Проверка правильности нового элемента
cmQuit	1	Завершение программы
cmError	2	Неопределено
cmMenu	3	Активность полосы меню
cmClose	4	Закрывает текущее окно
cmZoom	5	Масштабирует окно
cmResize	6	Изменяет размеры окна
cmNext	7	Делает активным следующее окно
cmPrev	8	Делает активным предыдущее окно

Стандартные команды TDialog

Команда	Значение	Назначение
cmOK	10	Нажата кнопка OK
cmCanced	11	Нажата кнопка Cancel или Esc

Команда	Значение	Назначение
cmYes	12	Нажата кнопка Yes
cmNo	13	Нажата кнопка No
cmDefault	14	Нажата кнопка по умолчанию или Enter

Режимы экрана

Константа	Значение	Назначение
smBW80	$0002	Черно-белый режим
smCO80	$0003	Цветной режим
smMono	$0007	Монохромный режим
smFont8x8	$0100	Режим 43 или 50 строк (EGA/VGA)

ПЕРЕМЕННЫЕ

Инициализированные переменные

Переменная	Тип	Начальное значение	Назначение
ButtonCount	Byte	0	Число кнопок мышки
MouseEvents	Boolean	False	Указывает на выбор мышкой
DoubleDelay	Word	8	Максимальное время задержки между двойными нажатиями
RepeatDelay	Word	8	Задержка между автоматическим повтором события от мышки

Неинициализированные переменные

Переменная	Тип	Назначение
MouseIntFlag	Byte	Только для внутреннего использования
MouseButtons	Byte	Какая кнопка была нажата
MouseWhere	TPoint	Позиция курсора мышки
StartupMode	Word	Режим экрана при запуске программы
ScreenMode	Word	Текущий режим экрана
ScreenWidth	Byte	Ширина экрана в колонках
ScreenHeight	Byte	Высота экрана в строках
CheckSnow	Boolean	Определяет «снежность» для CGA
HiResScreen	Boolean	Экран может отображать 43 или 50 строк (EGA/VGA)
ScreenBuffer	Pointer	Указатель на видеобуфер экрана
CursorLines	Word	Начало и окончание строк просмотра для установки типа курсора

Переменные обработчика системных ошибок

Переменная	Тип	Начальное значение	Назначение
SysErrorFunc	Sys Error Func	System Error	Функция, вызываемая монитором системной ошибки при ее возникновении
SysColorAttr	Word	$4E4F	Видеоатрибуты для сообщений об ошибках на цветном экране
SysMonoAttr	Word	$7070	Видеоатрибуты для сообщений об ошибках на монохромном экране
CtrlBreakHit	Boolean	False	Указывает
SaveCtrlBreak	Boolean	False	Статус проверки Ctrl-Break при запуске программы

ПРОЦЕДУРЫ И ФУНКЦИИ

Процедуры монитора событий

Процедура	Операция
InitEvents	Инициализирует монитор событий
DoneEvents	Закрывает монитор событий

Процедура	Операция
ShowMouse	Отображает курсор мышки
HideMouse	Стирает курсор мышки
GetMouseEvent	Создает запись события от мышки
GetKeyEvent	Создает запись события от клавиатуры

Процедуры управления экраном

InitVideo	Инициализирует монитор экрана
DoneVideo	Закрывает монитор экрана
SetVideoMode	Выбирает режим экрана (цветной, черно-белый, монохромный, высокого разрешения)
ClearScreen	Очищает экран при любом видео режиме

Функция по умолчанию обработчика системной ошибки

Функция	Операция
SystemError	Отображает сообщение об ошибке в нижней строке экрана и подсказки для завершения или повтора

Процедуры обработчика системной ошибки

Процедура	Операция
InitSysError	Инициализирует монитор системных ошибок
DoneSysError	Закрывает монитор системных ошибок

Функции поддержки клавиатуры

Функция	Операция
GetAltChar	Возвращает символ от клавиатуры
GetAltCode	Возвращает скэн-код от клавиатуры

Процедура форматирования строки

Процедура	Операция
FormatStr	Форматирует строку

Процедуры копирования буфера

Процедура	Операция
MoveBuf	Копирует буфер в другой буфер
MoveChar	Копирует одну или более копий символа в буфер
MoveCStr	Копирует строку управления в буфер
MoveStr	Копирует строку в буфер

Функция длины строки

Функция	Операция
CStrLen	Возвращает длину строки, игнорируя "~"

Инициализация драйвера

Процедура	Операция
InitDrivers	Инициализирует драйверы модуля

МОДУЛЬ TEXTVIEW

Модуль TextView содержит несколько специализированных видимых элементов для отображения текста в окне скроллинга.

ТИПЫ

Тип	Использование
TTerminal	TTY подобный текстового устройства
TTerminalBuffer	Круговой текстовый буфер для TTerminal
TTextDevice	Абстрактный объект текстового устройства

ПРОЦЕДУРА

Процедура	Операция
AssignDevice	Назначает устройство текстового файла для ввода и/или вывода

МОДУЛЬ MEMORY

Модуль Memory содержит процедуры монитора памяти Turbo Vision, которые обеспечивают функции управления кучей.

ПЕРЕМЕННЫЕ

Переменная	Тип	Начальное значение	Назначение
LowMemSize	Word	4096 div 16	Размер пула надежности

ПРОЦЕДУРЫ И ФУНКЦИИ

Процедура	Операция
InitMemory	Инициализирует монитор памяти
DoneMemory	Закрывает монитор памяти
GetBufMem	Распределяет кэш-буфер для группы
FreeBufMem	Удаляет кэш-буфер для группы
Функция	Операция
LowMemory	Указывает распределен ли буфер надежности
MemAlloc	Распределяет память с проверкой буфера надежности

МОДУЛЬ HISTLIST

Модуль HistList содержит все переменные, процедуры и функции необходимые для реализации списков истории.

ПЕРЕМЕННЫЕ

Переменная	Тип
HistoryBlock	Pointer
HistorySize	Word
HistoryUsed	Word

ПРОЦЕДУРЫ И ФУНКЦИИ

Процедура	Операция
HistoryAdd	Добавляет строку в список истории
ClearHistory	Очищает все списки истории
InitHistory	Инициализирует монитор списков истории
DoneHistory	Закрывает монитор списков истории
Функция	Операция
HistoryCount	Возвращает число строк в списке истории
HistoryStr	Возвращает отдельную строку из списка истории

Глава 3

СПРАВОЧНИК ПО ОБЪЕКТАМ

Эта глава содержит алфавитный список всех стандартных объектов Turbo Vision с объяснением их назначения и использования, с полями, методами и палитрами цветов.

Чтобы найти информацию по определенному объекту, помните, что многие свойства объектов в иерархии наследуются от предков. Вместо бесконечного дублирования всей информации эта глава описывает только поля и методы, которые добавляются или изменяются в этом объекте.

Например, если вы хотите найти поле Owner объекта TLabel, вы можете посмотреть поля TLabel, среди которых вы не найдете Owner. Затем посмотрите непосредственного предка TLabel в иерархии — TStaticText. Поля Owner нет опять. Посмотрите следующего непосредственного предка TView. Здесь вы найдете полную информацию об Owner, которое наследуется неизменным в TLabel.

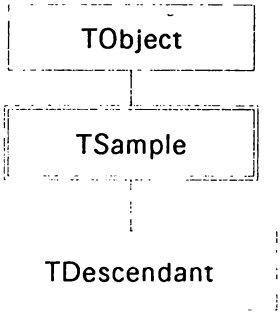
Каждый объект в этой главе имеет графическое представление предков и непосредственных наследников так, что вы сможете легко найти объекты, от которых наследуются поля и методы.

Каждый объект представлен в следующем формате:

Объект TSample — Модуль объекта

Поля

Этот раздел приводит список всех полей объекта. Кроме объявления поля и объяснения его использования приводится назначение «только чтение» и «чтение/запись». Поля «только на чтение» — это поля, которые устанавливаются и поддерживаются



методами объектов и которые не должны использоваться в левой части оператора присваивания.

AField AField: SomeType; Только чтение

AField — это поле, которое содержит некоторую информацию об этом объекте. Этот текст объясняет, как оно функционирует, что это означает и как вам его использовать.

См. также: Связанные поля, методы, объекты, глобальные функции и т.д.

AnotherField AnotherField: Word; Чтение/Запись

Это поле содержит информацию подобную информации для поля AField.

Методы

Этот раздел приводит все методы, которые либо определены в этом объекте, либо перекрывают унаследованные методы. Для виртуальных методов указывается, как часто требуется перекрывать метод: никогда, редко, иногда, часто или всегда.

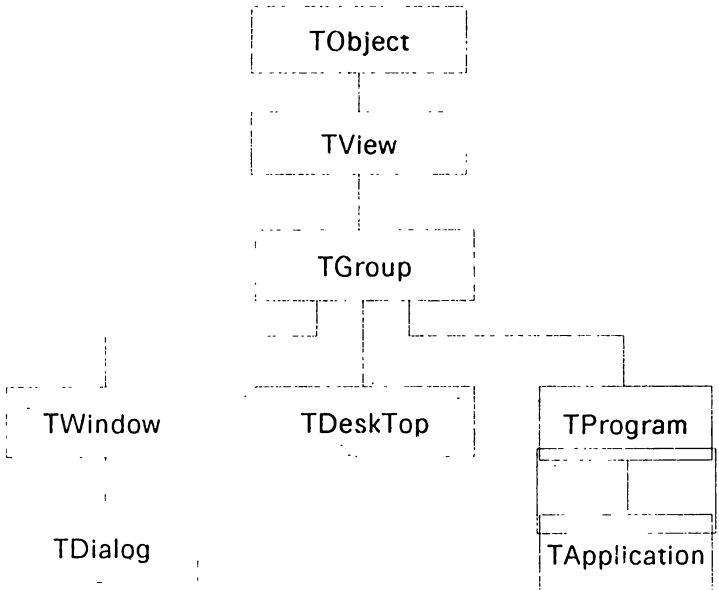
Init constructor Init(AParameter: SomeType);

Init создает новый экземпляр объекта, устанавливая поле AField в AParameter.

Zilch procedure Zilch; virtual; Перекрывается: Иногда

Процедура Zilch выполняет некоторые действия.

См. также TSomethingElse.Zilch



TApplication просто наследуется от TProgram и отличается от TProgram только конструктором и деструктором. TApplication.Init инициализирует все подсистемы Turbo Vision (управление памятью, видео, событиями, системными ошибками и списками историй) и затем вызывает TProgram.Init. Аналогично TApplication.Done вначале вызывает TProgram.Done, а затем уничтожает все подсистемы Turbo Vision.

Обычно вы будете наследовать свои программы от TApplication. Если вам потребуется другая последовательность инициализации подсистем и их закрытия, вы можете наследовать вашу программу от TProgram и вручную инициализировать и закрывать подсистемы Turbo Vision.

Методы

Init `constructor Init;`

Фактическая реализация TApplication.Init показана ниже:

```
constructor TApplication.Init;
begin
  InitMemory;
  InitVideo;
  InitEvents;
  InitSysError;
  InitHistory;
  TProgram.Init;
end;
```

См. также: TProgram.Init

Done `destructor Done; virtual;`

Реализация TApplication.Done показана ниже:

```
destructor TApplication.Done;
begin
  TProgram.Done;
  DoneHistory;
  DoneSysError;
  DoneEvents;
  DoneVideo;
  DoneMemory;
end;
```

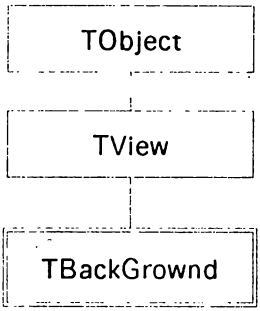
TBackground — App

TBackground — это простой видимый элемент, содержащий однотонно заполненный прямоугольник. Обычно он принадлежит TDeskTop.

Поля

Pattern `Pattern: Char;` — Только чтение

Это битовый шаблон для фона видимого элемента.



Методы

Init constructor Init(var Bounds: TRect; APattern: Char);
Создает объект TBackground с границами Bounds вызывая TView.Init. GrowMode устанавливается в gfGrowHiX + gfGrowHiY, а поле Pattern устанавливается в APattern.
См. также: TView.Init, TBackground.Pattern

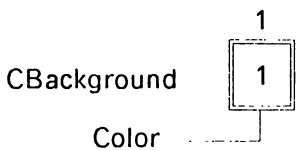
Load constructor Load(var S: TStream);
Создает объект TBackground и загружает его из потока S, вызывая TView.Load, а затем читая поле Pattern.
См. также: TView.Load

Draw procedure Draw; virtual; Перекрывается: Редко
Заполняет прямоугольник видимого элемента текущим шаблоном с цветом по умолчанию.

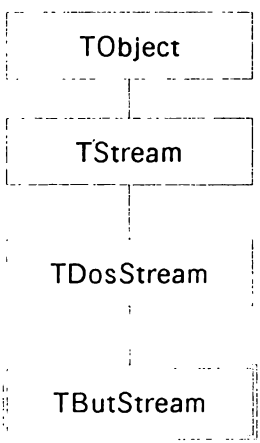
GetPalette function GetPalette: PPalette; virtual;
Перекрывается: Редко
Возвращает указатель на палитру по умолчанию CBackground.

Store procedure Store(var S: TStream);
Сохраняет видимый элемент TBackground в потоке, вызывая TView.Store, а затем записывая поле Pattern.
См. также: TView.Store, TBackground.Load

Объекты фона используют по умолчанию палитру CBackground для отображения в первый элемент палитры программы.



TBufStream — Objects



TBufStream реализует буферизованную версию TDosStream. Дополнительные поля указывают размер и положение буфера, а также текущую и последнюю позицию в буфере. Кроме перекрытия восьми методов TDosStream, TBufStream определяет абстрактный метод TStream.Flush. Конструктор TBufStream создает и открывает файл, вызывая TDosStream.Init, затем создает буфер с помощью GetMem. TBufStream значительно эффективнее TDosStream при работе большого числа небольших

данных в поток, а также при сохранении и загрузке объектов с использованием TStream.Get и TStream.Put.

Поля

Buffer Buffer: Pointer; Только чтение
Указатель на начало буфера потока.

BufSize BufSize: Word; Только чтение
Размер буфера в байтах.

BufPtr BufPtr: Word; Только чтение
Смещение от указателя Buffer, указывающее на текущую позицию внутри буфера.

BufEnd BufEnd: Word; Только чтение
Если буфер не заполнен, BufEnd даст смещение от указателя Buffer на последний используемый байт в буфере.

Методы

Init constructor Init(FileName: FNameStr; Mode, Size: Word);
Создает и открывает файл с режимом доступа Mode, вызывая TDosStream.Init. Так же создает буфер размером в SizeBuf, вызывая GetMem. Handle, Buffer и BufSize инициализируются соответственно. Типичный размер буфера от 512 до 2048 байт.
См. также: TDosStream.Init

Done destructor Done; virtual; Перекрывается: Никогда
Закрывает и освобождает файловый поток; выталкивает и освобождает его буфер.
См. также: TBufStream.Flush

Flush procedure Flush; virtual; Перекрывается: Никогда
Выталкивает буфер потока, обеспечивая, что поток будет в состоянии stOK.
См. также: TBufStream.Done

GetPos function GetPos: LongInt; virtual;
Перекрывается: Никогда

Возвращает значение текущей позиции потока (не перепутайте с BufPtr — текущей позицией в буфере).
См. также: TBufStream.Seek

GetSize function GetSize: LongInt; virtual;

Перекрывается: Никогда

Выталкивает буфер, а затем возвращает общее число байт в потоке.

Read procedure Read(var Buf; Count: Word); virtual;

Перекрывается: Никогда

Если stOK, читает Count байт в буфер Buf, начиная с текущей позиции потока.
Заметим, что Buf — это не буфер потока, а внешний буфер, содержащий данные читаемые из потока.
См. также: TBufStream.Write, stReadError

Seek procedure Seek(Pos: LongInt); virtual;

Перекрывается: Никогда

Выталкивает буфер, а затем устанавливает текущую позицию в Pos байт от начала потока. Начальная позиция потока — 0.
См. также: TBufStream.GetPos,
TBufStream.GetSize

Truncate procedure Truncate; virtual;

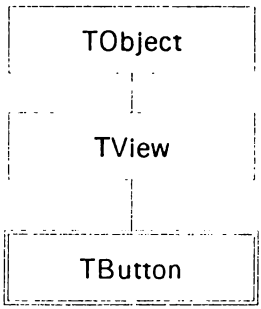
Перекрывается: Никогда

Выталкивает буфер, затем удаляет все данные потока от текущей позиции до конца потока. Текущая позиция устанавливается в новый конец потока.
См. также: TbufStream.GetPos, TBufStream.Seek

Write procedure Write(var Buf; Count: Word); virtual;

Перекрывается: Никогда

Если stOK, записывает Count байт из буфера Buf в поток, начиная с текущей позиции.
Заметим, что Buf — это не буфер потока, а внешний буфер, содержащий данные, записываемые в поток. Когда Write вызывается, Buf указывает на переменную, чье значение записывается.
См. также: TBudStream.Read, stWriteError



Объект TButton — это прямоугольник с заголовком и тенью, генерирующий команду при нажатии. Эти кнопки интенсивно используются в IDE. Кнопка может быть выбрана нажатием подсвеченной буквы, переходом на кнопку с помощью Tab и нажатием пробела, нажатием Enter, когда кнопка по умолчанию (указывается подсветкой) или отметкой кнопки мышкой.

При цветной и черно-белой палитрах кнопка имеет трехмерный вид, который изменяется при нажатии. На монохромных системах кнопка выделена стрелками и другие ASCII символы используются для указания, является ли кнопка по умолчанию выбранной и т.д.

Как и другие элементы управления, определенные в модуле Dialogs, TButton — это терминальный объект. Он может быть вставлен в любую группу и использован без перекрытия его методов.

Кнопка инициализируется передачей ей TRect строки заголовка, команды, генерируемой при нажатии кнопки и байта флагов. Чтобы определить для кнопки клавишу короткого набора, строка заголовка может содержать "~" вокруг одного из символов, который становится символом короткого набора. Параметр AFlag указывает, будет заголовок центрироваться или выравниваться по левой грани-

це и должна ли кнопка быть умалчиваемой (и следовательно выбираться через Enter).

Вы можете установить в окне или диалоговом окне только одну кнопку по умолчанию в любой момент времени. Кнопки, которые равны в группе, получают и отдают умалчиваемое состояние через сообщения evBroadcast. Кнопки могут быть разрешены или запрещены с использованием методов SetState и CommandEnabled.

Поля

Title Title: PString; Только чтение

Указатель на текст кнопки.

Command Command: Word; Только чтение

Слово команды в событии, генерируемом при нажатии кнопки. См. также: TButton.Init, TButton.Load

Flags Flags: Byte; Чтение/Запись

Flags — это поле, используемое для указания, будет ли текст кнопки центрироваться или выравниваться влево. Отдельные флаги описаны в разделе «Константы флага кнопки bfXXXX» главы 4.

См. также: TButton.Draw, константы bfXXXX

AmDefault AmDefault: Boolean; Только чтение

Если True, кнопка — по умолчанию (и следовательно выбирается при нажатии Enter). Иначе — это «нормальная» кнопка.

См. также: Константы флага кнопки bfXXXX

Методы

Init constructor Init(var Bounds: TRect; ATitle: TTitleStr; ACommand: Word; AFlags: Byte);

Создает объект TButton с заданным размером, вызывая TView. Init. Вызывается NewStr(ATitle) и назначается в Title. AFlags используется в двух целях: если AFlags and bfDefault — не 0, то AmDefault устанавливается в True; кроме того, AFlags указывает, будет заголовок центрироваться или выравниваться влево проверкой если AFlags and bfLeftJust не 0.

Options устанавливается в (ofSelectable + ofFirstClick + ofPreProcess + ofPostProcess). EventMask устанавливается в evBroadcast. Если данная ACommand не разрешена, в поле State устанавливается sfDisabled.
См. также: TView.Init, константы флага кнопки bfXXXX

Load constructor Load(var S: TStream);
Создает объект TButton и инициализирует его из заданного потока, вызывая TView.Load(S). Другие поля устанавливаются через вызовы S.Read, а State устанавливается в соответствии с тем, разрешена ли команда в поле Command. Используется совместно с TButton.Store для сохранения и получения объекта в TButton из TStream.
См. также: TView.Load, TButton.Store

Done destructor Done; virtual; Перекрывается: Никогда
Освобождает память, распределенную под Title, затем вызывает TView.Done для разрушения видимого элемента.
См. также: TView.Done

Draw procedure Draw; virtual; Перекрывается: Редко
Рисует кнопку соответствующей палитрой для ее текущего состояния (нормальная по умолчанию запрещена) и позиционирует метку в соответствии с битом bfLeftJust поля Flags.

GetPalette function GetPalette: PPalette; virtual;
Перекрывается: Иногда
Возвращает указатель на палитру по умолчанию CButton.

HandleEvent procedure HandleEvent(var Event: TEvent); virtual;
Перекрывается: Иногда
Отвечает на нажатие одним из трех способов: отметкой кнопки мышкой, нажатием короткой клавиши или становится кнопкой по умолчанию, когда получено общее сообщение cmDefault. Когда кноп-

ка нажата, генерируется командное событие с TView.PutEvent, с полем TButton.Command: установленным в Event.Command, а Event.InfoPtr установленным в @Self.

Кнопки также распознают общие команды cmGrabDefault и cmReleaseDefault, чтобы стать или «не стать» кнопкой по умолчанию и cmCommandSetChanged, которая заставляет их проверять, разрешены команды или запрещены.

См. также: TView.HandleEvent

MakeDefault procedure MakeDefault(Evable: Boolean);

Этот метод ничего не делает, если кнопка уже умалчиваемая. Иначе говорит Owner в кнопке измениться в состояние по умолчанию. Если Enable True, выдается общая команда cmGrebDefault, иначе cmReleaseDefault. Кнопка перерисовывается, чтобы показать новый статус.

См. также: TButton.AmDefault, bfDefault

SetState procedure SetState(AState: Word; Enable: Boolean); virtual;

Перекрывается: Редко

Вызывает TView.SetState, затем рисует кнопку, если кнопка стала sfSelected или sfActive. Если она стала активной (т. е. если AState — sfFocused) кнопка забирает или отдает состояние по умолчанию кнопке по умолчанию, вызывая MakeDefault.

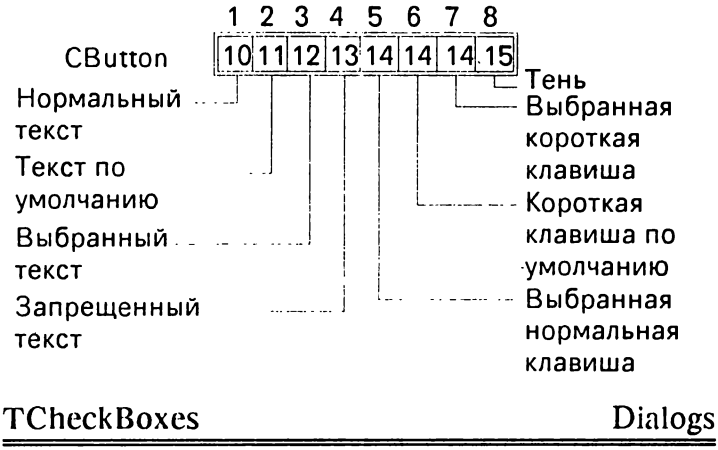
См. также: TCView.SetState, TButton.MakeDefault

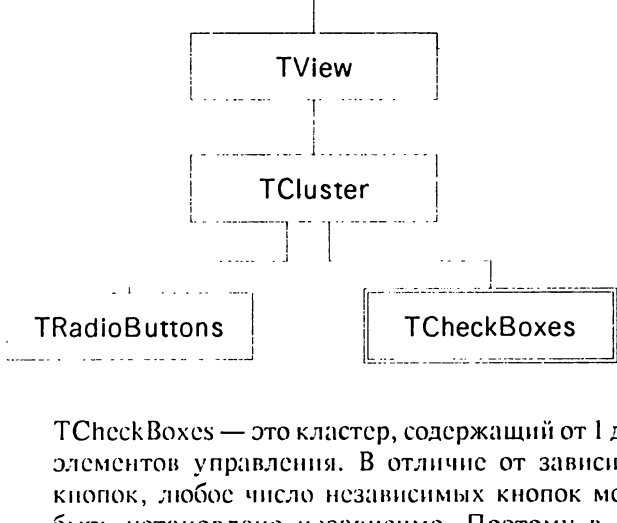
Store procedure Store(var S: TStream);

Сохраняет объект TButton в потоке, вызывая TView.Store(S), а затем S.Write для сохранения значений Title и Command. Используется совместно с TButton.Load для сохранения и получения объектов TButton из потока. См. также: TView.Store, TButton.Load, TStream.Write

Палитра

Объект кнопка использует палитру по умолчанию CButton для отображения элементов от 10 до 15 в палитру CDialog.





TCheckBoxes — это кластер, содержащий от 1 до 16 элементов управления. В отличие от зависимых кнопок, любое число независимых кнопок может быть установлено независимо. Поэтому в этой группе нет кнопки по умолчанию. Кнопки можно отмечать мышкой, движением курсора и коротким нажатием "Alt-буква". Каждая кнопка может быть подсвечена и включена/выключена (пробелом).

Если кнопка выбрана, появляется X. Другие части вашей программы обычно проверяют состояние независимых кнопок для определения, какая опция выбрана пользователем (например в IDE опции компилятора и редактора выбираются таким способом). Кластеры независимых кнопок часто связаны с объектами TLabel.

Поля

Наследуют поля ValueSl от TCluster. Value интерпретируется как набор из 16 бит (от 0 до 15), где 1 в бите позиции означает, что соответствующий элемент отмечен.

Методы

Заметим, что TCheckBoxes не перекрывает конструкторов, деструктора и обработчика событий TCluster. Порожденные типы объектов могут, однако, перекрыть их.

Draw procedure Draw; virtual; Перекрывается: Редко
Рисует объект TCheckBoxes вызывая наследуемый метод TCluster.DrawBox. По умолчанию независимая кнопка имеет вид: "[ ]", когда не выбрана, и "[X]", когда выбрана. Заметим, что если границы видимого элемента достаточно велики, независимые кнопки могут отображаться в несколько колонок.
См. также: TCluster.DrawBox

Mark function Mark(Item: Integer) : Boolean; virtual;
Перекрывается: Редко
Возвращает True, если бит элемента в Value установлен, т.е. если данная кнопка отмечена. Вы можете перекрыть это, установив другую интерпретацию поля Value. По умолчанию элементы нумеруются от 0 до 15.
См. также: TCheckBoxes.Press

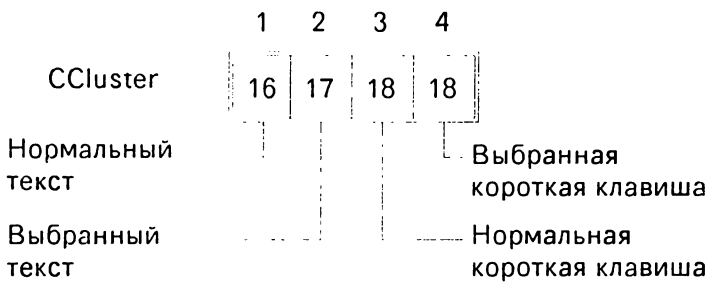
Press procedure Press(Item: Integer); virtual;
Перекрывается: Редко

Устанавливает бит элемента в Value. Вы можете перекрыть его для другой интерпретации поля Value. По умолчанию элементы нумеруются от 0 до 15.

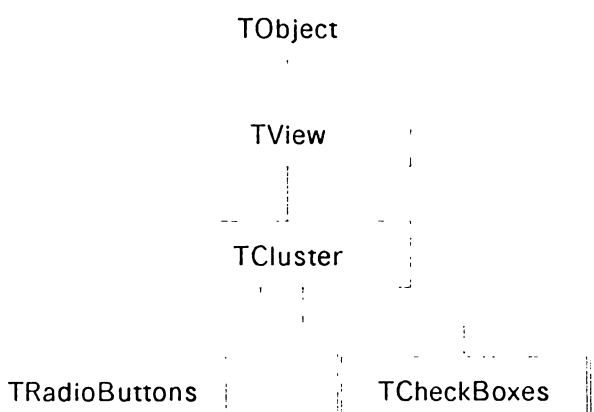
См. также: TCheckBoxes.Mark

Палитра

По умолчанию объекты независимых кнопок используют CCluster — палитру по умолчанию для всех объектов-кластеров.



TCluster — Dialogs



Кластер — это группа элементов управления, которые откликаются одинаково. TCluster — это абстрактный тип объекта, из которого порождаются группы элементов управления TRadioButtons и TCheckBoxes. Элементы управления кластера часто ассоциируются с объектами TLabel, позволяя вам выбирать элемент управления, выбирая дополнительную метку с объяснением. В то время, как кнопки используются для генерации команд, а строки ввода — для редактирования строк, кластеры используются для переключения битовых значений поля Value (типа Word). Стандартные наследники TCluster используют различные алгоритмы изменения Value: TCheckBoxes просто переключает бит, а TRadioButtons включает один бит и очищает предварительно выбранный бит. Оба объекта наследуют почти все свое поведение от TCluster.

Поля

Value — Value: Word; — Только чтение

Текущие значения элемента управления. Действительный смысл этого поля определяется методами, разработанными в типах объектов, порожденных от TCluster.

Sel — Sel: Integer; — Только чтение

Текущий выбранный элемент кластера.

Strings — Strings: TStringCollection; — Только чтение

Список элементов кластера.

Методы

Init — constructor Init(var Bounds: TRect; AStrings: PSItem);

Очищает поля Value и Sel. Параметр AStrings обычно выполняет серию вложенных вызовов глобальной функции NewSItem. Таким образом, весь кластер зависимых или независимых кнопок может быть создан одним вызовом конструктора:

```
var
  Control: PView;
...
R.Assign(30, 5, 52, 7);
Control := New(PRadioButtons, Init(R,
  NewSItem('~F~orward',
  NewSItem('~B~ackward', nil))));
...
```

Когда в кластер добавляются дополнительные зависимые или независимые кнопки, просто копируется первый вызов NewSItem и заголовок заменяется требуемым текстом. Затем добавляется дополнительная закрывающая скобка для каждой новой добавленной строки, и оператор будет компилироваться без синтаксических ошибок.

См. также: тип TSItem

Load constructor Load(var S: TStream);

Создает объект TCluster, вызывая TView.Load(S), затем устанавливает поля Value и Set вызовом S.Read. Наконец поле String кластера загружается из S с помощью Strings.Load(S). Используется совместно с TCluster.Store для сохранения и получения объектов TCluster из потока.

См. также: TCluster.Store, TView.Load

Done destructor Done; virtual; Перекрывается: Иногда

Освобождает память, распределенную под строку кластера, затем разрушает видимый элемент, вызывая TView.Done.

См. также: TView.Done

DataSize function DataSize: Word; virtual;

Перекрывается: Редко

Возвращает размер Value. Должен перекрываться в порожденных типах объектов, которые изменяют Value или добавляют другие поля данных для того, чтобы работать с GetData и SetData.

См. также: TCluster.GetData, TCluster.SetData

DrawBox procedure DrawBox(Icon: String; Maker: Char);

Вызывается методом Draw порожденного типа, чтобы рисовать прямоугольник перед строкой для каждого элемента кластера. Icon — это строка из 5 символов (' [ ] ' для независимых и ' ( ) ' для зависимых кнопок). Maker — это символ, используемый для указания, что кнопка отмечена ('X' для зависимых и '.' для независимых кнопок).

См. также: TCheckBoxes.Draw, TRadioButtons.Draw

GetData procedure GetData(var Rec); virtual;

Перекрывается: Редко

Записывает поле Value в данную запись и рисует кластер. Должен перекрываться в порожденных типах объектов, которые изменяют поле Value для того, чтобы работать с DataSize и SetData.

См. также: TCluster.DataSize, TCluster.SetData, TView.DrawView

GetHelpCtx function GetHelpCtx: Word; virtual;

Перекрывается:Редко

Возвращает значение Sel, добавленное к HelpCtx. Это позволяет Вам задать отдельную контекстную справку для каждого элемента кластера. Допустимый диапазон контекстов равен HelpCtx плюс число элементов кластера минус 1.

GetPalette function GetPalette: PPalette; virtual;

Перекрывается: Иногда

Возвращает указатель на палитру по умолчанию CCLuster.

HandleEvent procedure HandleEvent(var Event: TEvent); virtual;

Перекрывается: Редко

Вызывает TView.HandleEvent, который обрабатывает все события от мышки и клавиатуры, относящиеся к этому кластеру. Элементы управления выбираются отметкой мышки или клавишами движения курсора (включая пробел). Кластер

перерисовывается, чтобы показать выбранные элементы.

См. также: TView.HandleEvent

Mark function Mark(Item: Integer): Boolean; virtual;

Перекрывается: Всегда

Вызывается из Draw для определения, какие элементы отмечены. По умолчанию TCluster.Mark возвращает False. Mark должен перекрываться, возвращая True, если элемент управления в кластере отмечен, иначе False.

MovedTo procedure MovedTo(Item: Integer); virtual;

Перекрывается: Редко

Вызывается из HandleEvent для перемещения полосы выбора на заданный элемент управления в кластере.

Press procedure Press(Item: Integer); virtual;

Перекрывается: Всегда

Вызывается из HandleEvent когда элемент управления в кластере нажат либо отметкой мышки, либо событием от клавиатуры. Этот абстрактный метод должен быть перекрыт.

SetData procedure SetData(var Rec); virtual;

Перекрывается: Редко

Читает поле Value из данной записи и перерисовывает кластер. Должен перекрываться в порожденных типах кластеров, которые требуют другие поля для работы с DataSize и GetData.

См. также: TCluster.DataSize, YCluster.GetData, TView.DrawView

SetState procedure SetState(AState: Word;
Enable: Boolean); virtual;

Перекрывается: Редко

Вызывает TView.SetState, затем рисует кластер, если AState — sfSelected.

См. также: TView.SetState, TView.DrawView

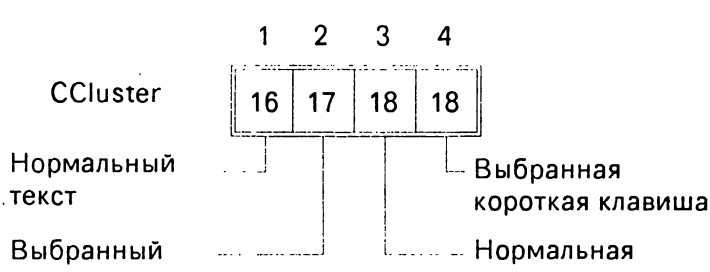
Store `procedure Store(var S: TStream);`

Сохраняет объект TCluster в потоке, вызывая TView.Store(S), записывает Value и Sel, затем сохраняет поле Strings кластера, используя его метод Store. Используется совместно с TCluster.Load для сохранения и получения объектов TCluster из потока.

См. также: TCluster.Load, TStream.Write

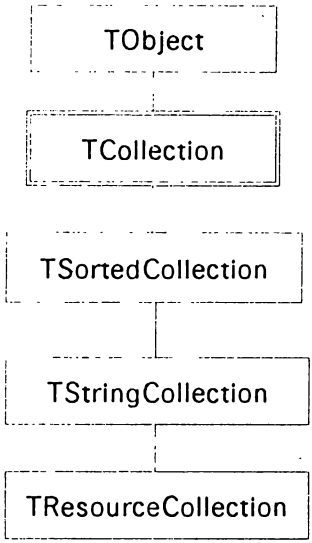
Палитра

Объекты TCluster используют CCluster — палитру по умолчанию для всех объектов кластера, чтобы отобразить элементы с 16 по 18 в палитру стандартного диалогового окна:



TCollection — Objects

TCollection — это абстрактный тип для реализации любой коллекции элементов, включая другие объекты. TCollection — это намного более общая концепция, чем обычные массив, множество или список. Размер объектов TCollection динамически устанавливается во время выполнения, и TCollection — базовый тип для многих специализированных типов, таких, как TSortedCollection, TStringCollection и TResourceCollection. В дополнение к методам добавления и удаления элементов TCollection представляет несколько итераторных



программ, которые вызывают процедуру или функцию для каждого элемента коллекции.

Поля

Items	Items: PItemList;	Только чтение
	Указатель на массив указателей элементов. См. также: тип TItemList	
Count	Count: Integer;	Только чтение
	Текущее число элементов в коллекции, максимально MaxCollectionSize. См. также: переменная MaxCollectionSize	
Limit	Limit: Integer;	Только чтение
	Текущий распределенный размер (в элементах) списка Items. См. также: Delta, TCollection.Init	
Delta	Delta: Integer;	Только чтение
	Число элементов, на которое увеличивается список Items при заполнении. Если Delta = 0, коллекция не может расти выше размера, установленного в Limit.	

Примечание. Увеличение размера коллекции достаточно дорого в смысле производительности. Чтобы минимизировать число раз когда это происходит, попытайтесь установить начальное Limit в такое количество, которое будет достаточно для всех элементов, которые вы собираетесь коллекционировать, и установите Delta так, чтобы она позволяла расширение на приемлемое количество.
См. также: Limit, TCollection.Init

Методы

Init — constructor Init(ALimit, ADelta: Integer);
Создает коллекцию с Limit, установленным в ALimit и Delta, установленным в ADelta. Первоначальное число элементов будет ограничено ALimit, но коллекция может расширяться, увеличиваясь по ADelta до тех пор, пока будет достаточно памяти или пока число элементов не достигнет MaxCollectionSize.
См. также: TCollection.Limit, TCollection.Delta.

Load — constructor Load(var S: TStream);
Создает и загружает коллекцию из потока. TCollection.Load вызывает GetItem для каждого элемента коллекции.
См. также: TCollection.GetItem

Done — destructor Done; virtual; Перекрывается: Часто
Удаляет и освобождает все элементы коллекции, вызывая TCollection.FreeAll и устанавливая Limit в 0.
См. также: TCollection.FreeAll, TCollection.Init

At — function At(Index: Integer) : Pointer;
Возвращает указатель на элемент с индексом Index в коллекции. Этот метод позволяет вам интерпретировать коллекцию как индексированный массив. Если индекс меньше 0 или больше или равно Count, вызывается метод Error с аргументом coIndexError и возвращается значение nil.
См. также: TCollection.IndexOf

AtDelete procedure AtDelete(Index: Integer);

Удаляет элемент в позиции Index и перемещает следующие элементы на одну позицию вверх. Count уменьшается на 1, но память, распределенная под коллекцию (как задано в Limit) не сокращается. Если Index меньше 0 или больше или равно Count, вызывается метод Error с аргументом coIndexError.

См. также: TCollection.FreeItem, TCollection.Free, TCollection.Delete

AtInsert procedure AtInsert(Index: Integer; Item: Pointer);

Вставляет Item в позицию Index и передвигает следующие элементы на одну позицию вниз. Если Index меньше 0 или больше Count, вызывается метод Error с аргументом coIndexError и новый Item не вставляется. Если Count равен Limit до вызова AtInsert, распределенный размер коллекции расширяется на Delta элементов, вызывая SetLimit. Если вызов SetLimit не может расширить коллекцию, вызывается метод Error с аргументом coOverflow и новый Item не вставляется.

См. также: TCollection.At, TCollection.AtPut

AtPut procedure AtPut(Index: Integer; Item: Pointer);

Заменяет элемент в позиции Index элементом, заданным в Item. Если Index меньше 0 или больше или равно Count, вызывается метод Error с аргументом coIndexError.

См. также: TCollection.At, TCollection.AtInsert

Delete procedure Delete(Item: Pointer);

Удаляет элемент Item из коллекции. Эквивалентно AtDelete(IndexOf(Item)).

См. также: TCollection.AtDelete, TCollection.DeleteAll

DeleteAll procedure DeleteAll;

Удаляет все элементы из коллекции, устанавливая Count в 0.

См. также: TCollection.Delete, TCollection.AtDelete

Error procedure Error(Code, Info: Integer); virtual;

Перекрывается: Иногда

Вызывается, когда встречается ошибка коллекции. По умолчанию этот метод генерирует ошибку времени выполнения 212.

См. также: константы коллекции coXXXX

FirstThat function FirstThat(Test: Pointer) : Pointer;

FirstThat применяет булевскую функцию, заданную указателем на функцию Test, к каждому элементу коллекции до тех пор, пока Test не возвратит True. Результат — указатель на элемент, для которого Test возвращает True, или nil, если функция Test возвращает False для всех элементов. Test должна указывать на дальнюю локальную функцию, использующую только один параметр типа Pointer и возвращающую значение типа Boolean. Например

```
function Matches(Item: Pointer) : Boolean; far;
```

Функция Test не может быть глобальной функцией. Предполагая, что List типа TCollection, оператор

```
P := List.FirstThat(@Matches);
```

соответствует

```
I := 0;
while (I  List.Count) and not Matches(List.At(I)) do
Inc(I);
if I < List.Count then P := List.At(I) else P := nil;
```

См. также: TCollection.LastThat, TCollection.ForEach

ForEarch procedure ForEarch(Action: Pointer);

ForEach применяет действие, определенное процедурой, на которую указывает Action, для каждого элемента коллекции. Action должен указывать на локальную дальнюю процедуру, использующую один параметр типа Pointer. Например

```
function PrintItem(Item: Pointer);
```

Процедура Action не может быть глобальной процедурой. Если List типа TCollection, оператор

```
List.ForEach(@PrintItem);
```

соответствует

```
for I := 0 to List.Count - 1 do PrintItem(List.At(I));
```

См. также: TCollection.FirstThat, TCollection.LastThat

Free procedure Free(Item: Pointer);

Удаляет и освобождает Item. Эквивалентно

```
FreeItem(Item);
Delete(Item);
```

См. также: TCollection.FreeItem, TCollection.Delete

FreeAll procedure FreeAll;

Удаляет и освобождает все элементы коллекции.
См. также: TCollection.DeleteAll

FreeItem procedure FreeItem(Item: Pointer); virtual;

Перекрывается: Иногда

Метод FreeItem должен освобождать Item. По умолчанию TCollection.FreeItem предполагает, что Item — это указатель на объект, порожденный от TObject и поэтому вызывает деструктор Done:

```
if Item <> nil then dispopse(PObject(Item), Done);
```

FreeItem вызывается из Free и FreeAll, но никогда не должен вызываться прямо.
См. также: TCollection.Free, TCollection.FreeAll

GetItem function TCollection.GetItem(var S: TStream): Pointer; virtual;

Перекрывается: Иногда

Вызывается из TCollection.Load для каждого элемента коллекции. Этот метод может быть перекрыт, но не должен вызываться напрямую. По

умолчанию TCollection.GetItem предполагает, что элементы коллекции порождены от TObject и вызывает TString.Get для загрузки элемента:

```
GetItem := S.Get;
```

См. также: TStream.Get, TCollection.Load, TCollection.Store

IndexOf `function IndexOf(Item: Pointer): Integer; virtual;`

Перекрывается: Никогда

Возвращает индекс для Item. Преобразует операцию в TCollection.At. Если Item — не в коллекции, IndexOf возвращает –1.

См. также: TCollection.At

Insert `procedure Insert(Item: Pointer); virtual;`

Перекрывается: Никогда

Вставляет Item в коллекцию, перестраивая другие индексы, если необходимо. По умолчанию вставка производится в конец коллекции вызовом AtInsert(Count, Item);

См. также: TCollection.AtInsert;

LastThat `function LastThat(Test: Pointer): Pointer;`

LastThat применяет булевскую функцию, заданную указателем на функцию Test, к каждому элементу коллекции в обратном порядке до тех пор, пока Test не вернет True. Результат — указатель на элемент, для которого Test возвращает True, или nil, если функция Test возвращает False для всех элементов. Test должен указывать на дальнюю локальную функцию, использующую один параметр типа Pointer и возвращающую типа Boolean, например

```
function Patches(Item: Pointer): Boolean; far;
```

Функция Test не может быть глобальной функцией. Если List типа TCollection, оператор

```
P := List.LastThat(@Matches);
```

соответствует

```
I := List.Count - 1;
while (I >= 0) and Matches(List.At(I)) do Dec(I);
 if I >= 0 then P := List.At(I) else P := nil;
```

См. также: TCollection.FirstThat, TCollection.ForEach;

Pack Procedure Pack;

Удаляет все nil указатели в коллекции.

См. также: TCollection.Delete, TCollection.DeleteAll

PutItem procedure PutItem(var S: TStream; Item: Pointer); virtual;

Перекрывается: Иногда

Вызывается из TCollection.Store для каждого элемента коллекции. Этот метод может быть перекрыт, но не должен вызываться прямо. По умолчанию TCollection.PutItem предполагает, что элементы коллекций порождаются от TObject и вызов TString.Put сохраняет элемент:

```
S.Put(Item);
```

См. также: TCollection.GetItem, TCollection.Store, TCollection.Load

SetLimit procedure SetLimit(ALimit: Integer); virtual;

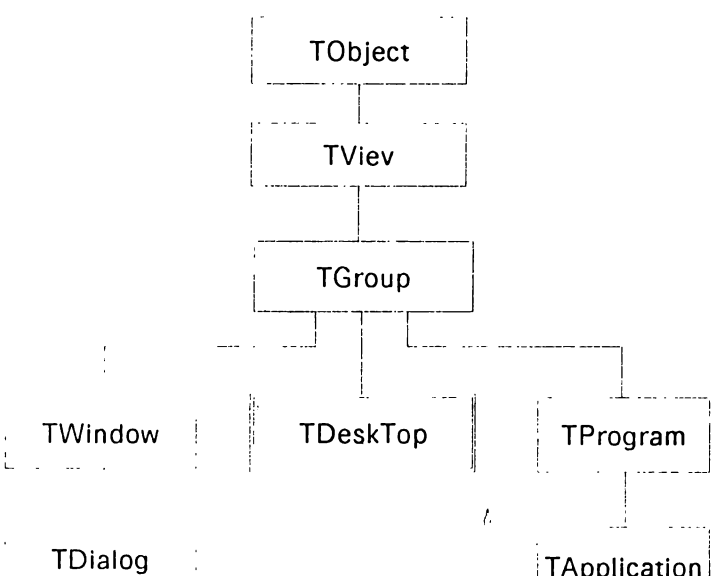
Перекрывается: Редко

Расширяет или сокращает коллекцию, изменяя распределенный размер в ALimit. Если ALimit меньше Count, он устанавливается в Count, и если ALimit больше MaxCollectionSize, он устанавливается в MaxCollectionSize. Кроме того, если ALimit отличается от текущего Limit, распределяется новый массив Items из Alimit элементов, старый массив Items копируется в новый массив и старый массив освобождается.

См.также: TCollection.Limit, TCollection.Count, переменная MaxCollectionSize

Store procedure Store(var S: TStream);
Сохраняет коллекцию и все ее элементы в потоке S. TCollection.Store вызывает TCollection.PutItem для каждого элемента коллекции.
См. также: TCollection.PutItem

TDeskTop App



TDescTop — это простая группа, которая владеет видимым элементом TBackground, на котором появляются окна и другие видимые элементы программы. TDeskTop представляет область панели экрана, располагаемой между верхней полосой меню и нижней строкой статуса.

Методы

Init constructor Init(var Bounds: TRect);
Создает группу TDeskTop с размером Bounds. По умолчанию GrowMode установлена в gfGrowHiX +

gfGrowHiY. Init также вызывает NewBackground для вставки видимого элемента TBackgroud в группу.

См. также: TDeskTop.NewBackGround, TGroup.Init, TGroup.Insert

Cascade procedure Cascade(var R: TRect);

Заново отображает все окна, принадлежащие панели экрана, в каскадном формате. Первое окно в Z-порядке (самое нижнее) расширяется на всю панель экрана, а каждое последующее окно заполняет область, начинающуюся на одну строку ниже и на одну колонку правее, чем предыдущее. Активное окно появляется на вершине как самое наименьшее.

См. также: ofTileable, TDeskTop.Tile

NewBackGround function NewBackGround: PView; virtual;

Перекрывается: Иногда

Возвращает указатель на фон, используемый в панели экрана. Этот метод вызывается в методе TDeskTop.Init. Наследуемые объекты могут изменить тип фона, перекрывая этот метод.

См. также: TDeskTop.Init

HandleEvent procedure HandleEvent(var Event: TEvent); virtual;

Перекрывается: Редко

Вызывает TGroup.HandleEvent и отрабатывает команды cmNext (обычно горячая клавиша F6) и cmPrevious циклическим проходом по окнам (начиная с текущего выбранного видимого элемента) принадлежащим панели экрана.

См. также: TGroup.HandleEvent, константы команд cmXXXX

Tile procedure Tile(var R: TRect);

Заново отображает все видимые элементы с ofTileable, принадлежащие панели экрана, в формате заполнения.

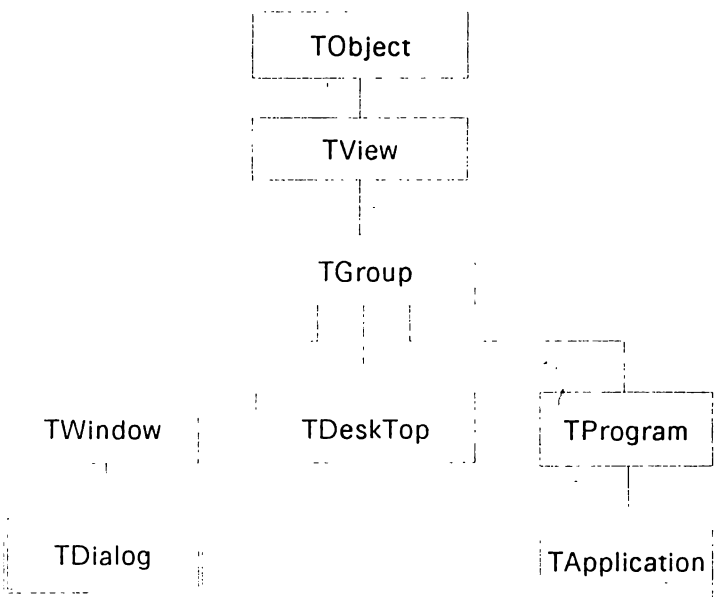
См. также: TDeskTop.Cascade, ofTileable

TileError procedure TileError; virtual;

Перекрывается: Иногда TileError вызывается, если возникла ошибка во время работы TDeskTop.Tile или TDeskTop.Cascade. По умолчанию ничего не делает. Вы можете перекрыть ее, чтобы указать пользователю, что программа не может реорганизовать окна.

См. также: TDeskTop.File, TDeskTop.Cascade

TDialog — Dialogs



TDialog — это потомок TWindow со следующими свойствами:

- GrowMode = 0, т.е. диалоговые окна не увеличиваются.
- Флаги wfMove и wfClose установлены, т.е. диалоговые окна можно перемещать и закрывать (предоставлена закрывающая кнопка).

- Обработчик событий TDialog вызывает TWindow.HandleEvent и дополнительно обрабатывает отклики на клавиши Esc и Enter. Клавиша Esc генерирует команду cmCancel, а Enter генерирует команду cmDefault.
- TDialog.Valid возвращает True на команду cmCancel, иначе вызывает TGroup.Valid.

Методы

Init `constructor Init(var Bounds: TRect; ATitle: TTitleStr);`

Создает диалоговое окно с заданным размером и заголовком, вызывая TWindow.Init(Bounds, ATitle, wnNoNumber). GrowMode устанавливается в 0 и Flags устанавливается в wfMove + wfClose. Это означает, что по умолчанию диалоговые окна можно перемещать и закрывать, но нельзя изменять их размеры.

Заметим, что TDialog не определяет собственного деструктора, а использует Close и Done, наследуемые через TWindow, TGroup и TView.

См. также: TWindow.Init

HandleEvent `procedure HandleEvent(var Event: TEvent); virtual;`

Перекрывается: Иногда

Вызывает TWindow.HandleEvent(Event), затем обрабатывает клавиши Enter и Esc. В частности, Esc генерирует команду cmCancel, а Enter посылает общие сообщения cmDeafult. Этот метод также обрабатывает cmOK, cmCancel, cmYes и cmNo, заканчивая модальное состояние диалогового окна. Для каждого из успешно обработанных событий он вызывает метод ClearEvent.

См. также: TWindow.HandleEvent

GetPalette `function GetPalette: PPalette; virtual;`

Перекрывается: Редко

Этот метод возвращает указатель на палитру по умолчанию CPalette.

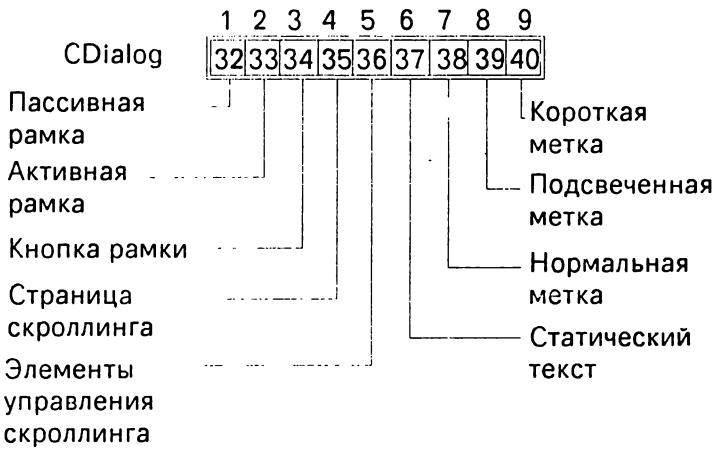
Valid function Valid(Command: Word): Boolean; virtual;

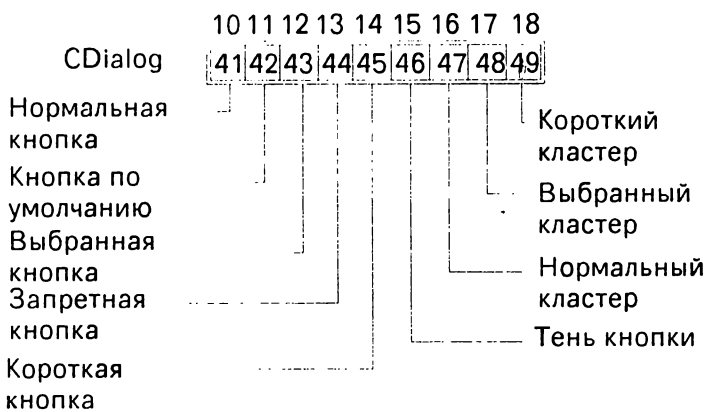
Перекрывается: Редко

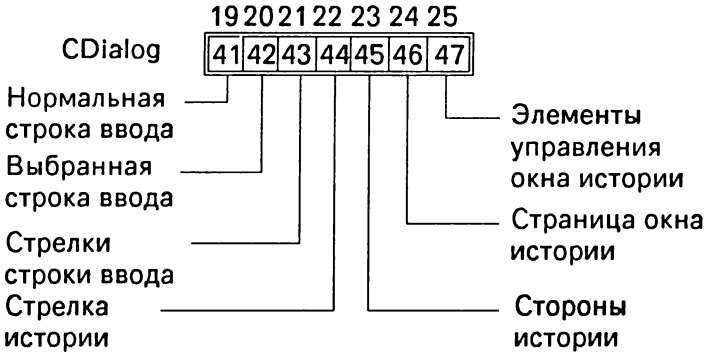
Возвращает True, если заданная команда — cmCancel, или если все элементы управления группы возвращают True.

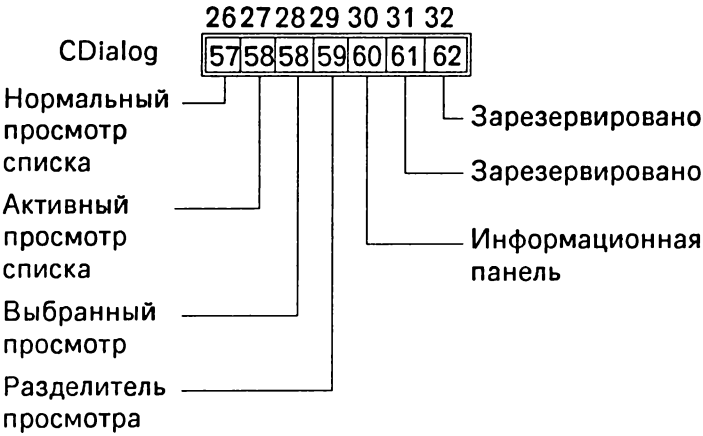
См. также: TGroup.Valid

Палитра



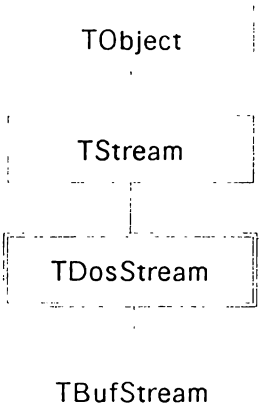






Объекты диалогового окна используют палитру по умолчанию CDialog для отображения с 32 по 63 элемент в палитру программы.

См. также: Метод GetPalette для каждого типа объекта.



TDosStream — это специализированный TStream, реализующий небуферизованный поток файла DOS. Конструктор позволяет вам создать или открыть файл DOS, задав его имя и режим доступа: stCreate, stOpenRead, stOpenWrite или stOpen. Добавляется поле Handle — обработчик традиционного файла DOS, используемый для доступа к открытому файлу. Большинство программ будут использовать буферизованный поток TBufStream, порожденный от TDosStream. TDosStream перекрывает все абстрактные методы TStream, за исключением TStream.Flush.

Поля

Handle: Word; Только чтение

Handle — это обработчик файла DOS используемый только для доступа к открытому файлу потока.

Методы

Init constructor Init(FileName: FNameStr; Mode: Word);

Создает поток файла DOS с именем FileName и заданным режимом доступа. Если успешно, поле Handle устанавливается в обработчик файла DOS.

Ошибка указывается вызовом Error с аргументом stInitError.
Аргумент Mode должен принимать одно из значений: stCreate, stOpenRead, stOpenWrite или stOpen. Эти константы объяснены в «Константы потока stXXXX» главы 4.

Done destructor Done; virtual; Перекрывается: Никогда
Закрывает и освобождает поток файла DOS
См. также: TDosStream.Init

GetPos function GetPos: Longint; virtual;
Перекрывается: Никогда
Возвращает значение текущей позиции в потоке.
См. также: TDosStream.Seek

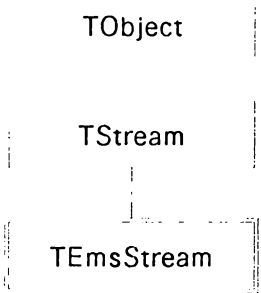
GetSize function GetSize: Longint; virtual;
Перекрывается: Никогда
Возвращает размер потока в байтах.

Read procedure Read(var Buf; Count: Word); virtual;
Перекрывается: Никогда
Читает Count байт в буфер Buf, начиная с текущей позиции потока.
См. также: TDosStream.Write, stReadError

Seek procedure Seek(Pos: Longint); virtual;
Перекрывается: Никогда
Устанавливает текущую позицию в Pos байт от начала потока.
См. также: TDosStream.GetPos, TDosStream.GetSize

Truncate procedure Truncate; virtual;
Перекрывается: Никогда
Удаляет все данные текущего потока от текущей позиции до конца потока.
См. также: TDosStream.GetPos, TDosStream.Seek

Write procedure Write(var Buf; Count: Word); virtual;
Пишет Count байт из буфера Buf в поток, начиная с текущей позиции.
См. также: TDosStream.Read, stWriteError



TEmsStream — это специализированный поток, реализующий поток в EMS памяти, порожденный от TStream. Дополнительные поля представляют обработчик EMS, число страниц, размер потока и текущую позицию. TStreamEms перекрывает 6 абстрактных методов TStream, а также предоставляет специальный конструктор и деструктор.

Примечание. При отладке программы, использующей EMS потоки, IDE не может восстановить EMS память, распределенную вашей программой, если ваша программа преждевременно завершилась или вы забыли вызвать деструктор Done для EMS потока. Только метод Done (или перезагрузка) могут освободить EMS страницы, принадлежавшие потоку.

Поля

Handle Handle: Word; Только чтение

Обработчик EMS для потока.

PageCount PageCount: Word; Только чтение

Число распределенных для потока страниц, по 16K на страницу.

Size Size: Longint; Только чтение

Размер потока в байтах.

Position Position: Longint; Только чтение

Текущая позиция внутри потока. Первая позиция — 0.

Методы

Init constructor Init(MinSize: Longint);

Создает EMS поток с заданным минимальным размером в байтах. Вызывает TStream.Init, затем устанавливает Handle, Size и PageCount. Вызывает Error с аргументом stInitError, если инициализация неудачна.

См. также: TEmsStream.Done

Done destructor Done; virtual; Перекрывается: Никогда

Освобождает EMS поток и используемые EMS страницы.

См. также: TEmsStream.Init

GetPos function GetPos: Longint; virtual;
Перекрывается: Никогда

Возвращает значение текущей позиции в потоке.

См. также: TEmsStream.Seek

GetSize function GetSize: Longint; virtual;
Перекрывается: Никогда

Возвращает общий размер потока.

Read procedure Read(var Buf; Count: Word); virtual;
Перекрывается: Никогда

Читает Count байт из буфера Buf, начиная с текущей позиции в потоке.

См. также: TEmsStream.Write, stReadError

Seek procedure Seek(Pos: Longint); virtual;
Перекрывается: Никогда

Устанавливает текущую позицию в Pos байт от начала потока.

См. также: TEmsStream.GetPos, TEmsStream.GetSize

Truncate procedure Truncate; virtual;

Перекрывается: Никогда

Удаляет все данные в потоке, начиная с текущей позиции до конца потока. Текущая позиция устанавливается в новый конец потока.

См. также: TDosStream.GetPos, TDosStream.Seek

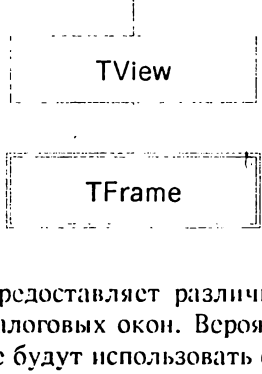
Write procedure Write(var Buf; Count: Word); virtual;

Перекрывается: Никогда

Записывает Count байт из буфера Buf в поток, начиная с текущей позиции.

См. также: TDosStream.Read, TEmsStream.GetPos, TEmsStream.Seek

TFrame — Views



TFrame предоставляет различные рамки вокруг окон и диалоговых окон. Вероятно, пользователи никогда не будут использовать объекты рамок напрямую, т.к. они добавляются к окнам по умолчанию.

Методы

Init constructor Init(var Bounds: TRect);

Вызывает TView.Init, затем устанавливает GrowMode в gfGrowHiX + gfGrowHiY и устанавливает EventMask в EventMask or evBroadcast, так,

чтобы объекты TFrame обрабатывали по умолчанию общие события.
См. также: TView.Init

Draw procedure Draw; virtual; Перекрывается: Редко
Рисует рамку с цветом атрибутов и кнопками, соответствующими текущим флагам State: активный, неактивный и перемещаемый. Добавляет кнопки изменения размера, закрытия и масштабирования в зависимости от Flags окна-владельца. Добавляет заголовок, если есть, из поля Title окна-владельца. Активные окна рисуются с двойной рамкой и кнопками, а неактивные окна с одинарной рамкой и без кнопок.
См. также: константы флагов состояния sfXXXX, константы флагов окна wfXXXX

GetPalette function GetPalette: Palette; virtual;
Перекрывается: Редко
Возвращает указатель на палитру рамки по умолчанию CFrame.

HandleEvent procedure HandleEvent(var Event: TEvent); virtual;
Перекрывается: Редко
Вызывает TView.HandleEvent, затем обрабатывает события от мышки. Если закрывающая кнопка отмечена мышкой, TFrame генерирует события cmClose. Отметка кнопки масштабирования или двойная отметка верхней линии рамки генерирует событие cmZoom. При перемещении мышкой за верхнюю строку рамки окно перемещается, а перемещение за кнопку изменения размера передвигает нижний правый угол элемента и соответственно изменяет его размер.
См. также: TView.HandleEvent

SetState procedure SetState(AState: Word;
Enable: Boolean); virtual;

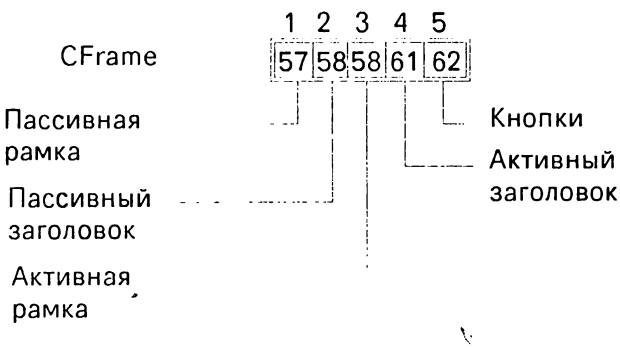
Перекрывается: Редко

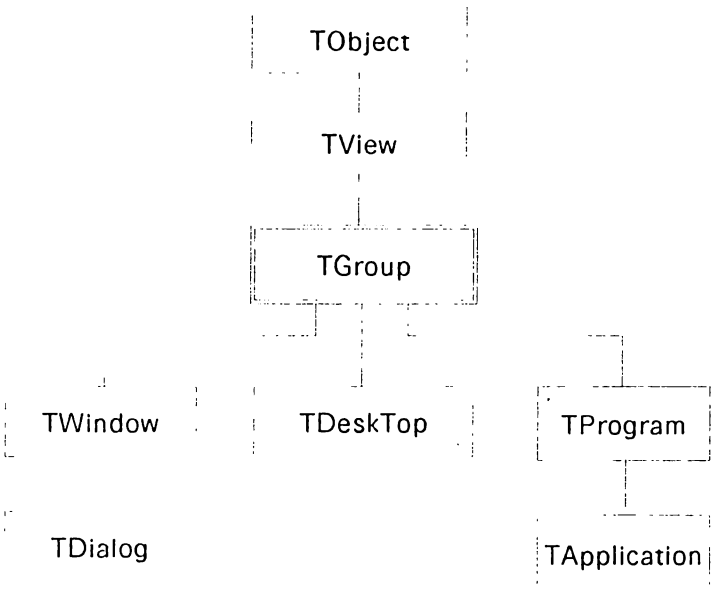
Вызывает TView.SetState, затем, если новое состояние — sfActive или sfDragging, вызывает DrawView для перерисовки видимого элемента.

См. также: TView.SetState

Палитра

Объекты рамок используют палитру по умолчанию CFrame для отображения первых трех элементов в палитру стандартного окна.





Объекты TGroup и их порождения (которые мы для краткости называем группами) — основная управляющая сила Turbo Vision. Группы — это специальные порождения видимых элементов. В дополнение ко всем полям и методам, порожденным от TView, группа имеет дополнительные поля и методы (включая перекрывающие), позволяющие управлять динамически связанными списками видимых элементов (включая другие группы) как одним объектом. Мы часто говорили о подэлементах группы даже если эти подэлементы сами являются группами.

Хотя группа имеет прямоугольную границу от своего предка TView, группа видима только через отображение своих подэлементов. Концептуально группа рисует себя через методы Draw своих подэлементов. Группа владеет своими подэлементами,

и они должны обеспечивать зарисовку (заполнение) всего прямоугольника группы Bounds. Во время работы программы подэлементы и подгруппы создаются, вставляются в группы и отображаются в результате деятельности пользователя и событий, генерируемых программой. Подэлементы могут быть легко скрыты, удалены из группы действиями пользователя (такими, как закрытие окна или выход из диалогового окна).

Три порожденных от TGroup типа: TWindow, TDeskTop и TApplication (через TProgram) иллюстрируют концепцию групп и подгрупп. TApplication обычно владеет объектами TDeskTop, TStatusLine и TMenuView. TDeskTop порожден от TGroup и таким образом может владеть объектами TWindow, которые в свою очередь владеют объектами TFrame, TScrollBar и т.д.

Объекты TGroup передают рисование и обработку событий своим подэлементам, как объясняется в главах 4 и 5 Описания. Многие из основных методов TView перекрываются в TGroup. Например, сохранение и чтение группы из потока может быть достигнуто одним вызовом TGroup.Store и TGroup.Load.

Объекты TGroup обычно не имеют экземпляров; вы будете создавать экземпляры от типов объектов, порожденных от TGroup: TApplication, TDeskTop и TWindow.

Поля

Last Last : PView; Только чтение

Указывает на последний подэлемент группы (самый дальний от вершины в Z-порядке). Поле Next последнего подэлемента указывает на первый подэлемент, чье поле Next указывает на следующий подэлемент и т.д., формируя циклический список.

Current Current: PView; Только чтение

Указывает на выбранный подэлемент или равен nil, если нет выбранного подэлемента.

См. также: sfSelected, TView.Select

Buffer — Buffer: PVideoBuf; — Только чтение

Указывает на буфер, используемый для кэширования операций перерисовки или равен nil, если группа не имеет кэш-буфера. Кэш-буфера создаются и уничтожаются автоматически, если флаг ofBuffered в поле Options не очищен.

См. также: TGroup.Draw, TGroup.Lock, TGroup.Unlock

Phase — Phase: (phFocused, phPreProcess, phPostProcess); — Только чтение

Текущая фаза обработки для активного события. Подэлементы, в которых установлены флаги ofPreProcess и/или ofPostProcess могут проверять Owner^.Phase, чтобы определить, в какой из фаз phPreProcess, phFocused или phPostProcess был вызван их HandleEvent.

См. также: ofPreProcess, ofPostProcess, TGroup.HandleEvent

Методы

Init — constructor Init(var Bounds: TRect);

Вызывает TView.Init, устанавливает в Options ofSelectable и ofBuffered и устанавливает EventMask в $FFFF.

См. также: TView.Init, TGroup.Load

Load — constructor Load(var S: TStream);

Загружает всю группу из потока, вызывая вначале наследуемый TView.Load, а затем используя TStream.Get для чтения каждого подэлемента. После загрузки всех подэлементов выполняет проход по подэлементам для установки всех считанных указателей с использованием GetPeerViewPtr.

Если тип объекта, порожденного от TGroup, содержит поля, которые указывают на подэлементы, он должен использовать GetSubViewPtr внутри Load, чтобы читать эти поля.

См. также: TView.Load, TGroup.Store, TGroup.GetSubViewPtr

Done destructor Done; virtual; Перекрывается: Часто
Перекрывает TView.Done. Скрывает группу, используя Hide, освобождает каждый элемент группы, используя Dispose(P, Done) и, наконец, вызывает наследуемый TView.Done.
См. также: TView.Done

ChangeBounds procedure ChangeBounds(var Bounds: Trect); virtual;
Перекрывается: Никогда
Перекрывает TView.ChangeBounds. Изменяет границы группы в Bounds, затем вызывает CalcBounds и ChangeBounds для каждого элемента группы.
См. также: TView.CalcBounds, TView.ChangeBounds

DataSize function DataSize: Word; virtual;
Перекрывается: Редко
Перекрывает TView.DataSize. Возвращает общий размер группы, вызывая и накапливая DataSize для каждого подэлемента.
См. также: TView.DataSize

Delete procedure Delete(P: PView);
Удаляет подэлемент P из группы и перерисовывает другие подэлементы, если необходимо. Поля Owner и Next в P устанавливаются в nil.
См. также: TGroup.Insert

Draw procedure Draw; virtual; Перекрывается: Никогда
Перекрывает TView.Draw. Если кэш-буфер существует (см. поле TGroup.Buffer), то буфер пишется на экран с использованием TView.WriteBuf. Иначе каждый подэлемент рисует себя с помощью TGroup.Redraw.
См. также: TGroup.Buffer, TGroup.Redraw

EndModal procedure EndModal(Command: Word); virtual;
Перекрывается: Никогда

Если группа — это текущий модальный видимый элемент, модальное состояние завершается. Command передается в ExecView, который возвращает Command как результат. Если эта группа не текущий модальный видимый элемент, она вызывает TView.EndModal.

См. также: TGroup.ExecView, TGroup.Execute

EventError procedure EventError(var Event: TEvent); virtual;

Перекрывается: Иногда EventError вызывается когда в цикле обработчика события модального TGroup.Execute встречается событие, которое не может быть обработано. Действие по умолчанию: если Owner группы не nil, EventError вызывает EventError своего владельца. Обычно эта цепочка распространяется до EventError из TApplication. Вы можете перекрыть EventError для выполнения требуемого действия.

См. также: TGroup.Execute, TGroup.ExecView, sfModal

ExecView function ExecView(P: PView): Word;

ExecView — это модальный вариант немодальных методов Insert и Delete. В отличие от Insert, после вставки видимого элемента в группу ExecView ожидает видимый элемент для выполнения, затем удаляет видимый элемент и возвращает результат выполнения. ExecView используется в ряде мест в Turbo Vision, например для реализации TApplication.Run и для выполнения модальных диалоговых окон.

ExecView сохраняет текущий контекст (выбранный видимый элемент, модальный видимый элемент и набор команд) делает P модальным, вызывая P^.SetState(sfModal, True), вставляет P в группу (если он еще не вставлен) и вызывает P^.Execute. Когда P^.Execute возвращает управление, группа восстанавливается в предыдущее состояние и результат P^.Execute возвращается как результат

вызова ExecView. Если P — nil в вызове ExecView, возвращается значение cmCancel.

См. также: TGroup.Execute, sfModal

Execute function Execute: Word; virtual;

Перекрывается: Редко

Перекрывает TView.Execute. Execute — это главный цикл событий группы: она постоянно получает события, используя GetEvent и обрабатывает их, используя HandleEvent. Цикл событий завершается группой или подэлементом с помощью вызова EndModal. Однако до возврата Exec вызывает Valid для проверки, что модальное состояние в самом деле было завершено.

Реализация TGroup.Execute показана ниже. Заметим, что EndState — это private поле в TGroup, которое устанавливается вызовом EndModel.

```
function TGroup.Execute: Word;
var
  E: TEvent;
begin
  repeat
    EndState := 0;
    repeat
      GetEvent(E);
      HandleEvent(E);
      if E.What <> evNothing then EventError(E);
    until EndState <> 0;
  until Valid(EndState);
  Execute := EndState;
end;
```

См. также: TGroup.GetEvent, TGroup.HandleEvent, TGroup.EndModal, TGroup.Valid

First function First: PView;

Возвращает указатель на первый подэлемент (ближайший к вершине в Z-порядке) или nil, если в группе нет подэлементов.

См. также: TGroup.Last

FirstThat function FirstThat(Test: Pointer): PView;

FirstThat применяет булевскую функцию, заданную указателем на функцию Test, применительно к каждому подэлементу в Z-порядке до тех пор, пока Test не вернет True. Результат — указатель на подэлемент, для которог Test возвращает True, или nil, если функция Test возвращает False для всех подэлементов. Test должна указывать дальнюю локальную функцию, использующую параметр типа Pointer и возвращающую значение типа Boolean. Например:

```
function MyTestFunc(P: PView): Boolean; far;
```

Метод SubViewAt, показанный ниже, возвращает указатель на первый подэлемент, содержащий данную точку.

```
function TMyGroup.SubViewAt(Where: TPoint):
                                    PView;

  function ContainsPoint(P: PView): Boolean; far;
  var
    Bounds: TRect;
  begin
    P^.GetBounds(Bounds);
    ContainsPoint := (P^.State and sfVisible <> 0)
      and Bounds.Contains(Where);
  end;

begin
  SubViewAt := FirstThat(@ContainsPoint);
end;
```

См. также: TGroup.ForEach

ForEach procedure ForEach(Action: Pointer);

ForEach выполняет действие, заданное указателем Action на процедуру, применительно к каждому подэлементу группы в Z-порядке. Action должна

указывать на дальнюю локальную процедуру, использующую параметр типа Pointer, например:

```
procedure MyActionProc(P: PView); far;
```

Метод MoveSubViews перемещает все подэлементы группы на значение, заданное в Delta. Заметим использование Lock и UnLock для ограничения числа выполняемых операций по перерисовке для предотвращения неприятного мерцания.

```
procedure TMyGroup.MoveSubViews(Delta: TPoint);

  procedure DoMoveView(P: PView); far;
  begin
    P^.MoveTo(P^.Origin.X + Delta.X, P^.Origin.Y
    + Delta.Y);
  end;

begin
  Lock;
  ForEach(@DoMoveView);
  UnLock;
end;
```

См. также: TGroup.FirstThat

GetData `procedure GetData(var Rec); virtual;`

Перекрывается: Редко

Перекрывает TView.GetData. Вызывает GetData для каждого подэлемента в Z-порядке, увеличивая положение, заданное в Rec, на DataSize для каждого подэлемента.

См. также: TView.GetData, TGroup.SetData

GetHelpCtx `function GetHelpCtx: Word; virtual;`

Перекрывается: Редко

Возвращает контекст подсказки для текущего активного видимого элемента, вызывая метод выбранного подэлемента GetHelpCtx. Если нет контекста подсказки, заданного подэлементом, GetHelpCtx возвращает значение собственного поля HelpCtx.

GetSubViewPtr procedure GetSubViewPtr(var S: TStream; var P);

Загружает указатель на подэлемент P из потока S. GetSubViewPtr должна использоваться только внутри конструктора Load для чтения значений указателей, которые были записаны вызовом PutSubViewPtr из метода Store.

См. также: TView.PutSubViewPtr, TGroup.Load, TGroup.Store

HandleEvent procedure HandleEvent(var Event: TEvent); virtual;

Перекрывается: Часто

Перекрывает TView.HandleEvent. Группа обрабатывает события, передавая их в методы HandleEvent одного или более подэлементов. Однако действительный маршрут зависит от класса события.

Для активных событий (по умолчанию evKeyDown и evCommand, см. переменную FocusedEvents) обработка событий выполняется в три фазы: во-первых, поле Phase устанавливается в phPreProcess, и событие передается в HandleEvent всех подэлементов, в которых установлен флаг ofPreProcess. Затем Phase устанавливается в phFocused, и событие передается в HandleEvent текущего выбранного видимого элемента. Наконец, Phase устанавливается в phPostProcess, и событие передается в HandleEvent всех подэлементов, в которых установлен флаг ofPostProcess. Для позиционированных событий (по умолчанию evMouse, см. переменную PositionalEvents) событие передается в HandleEvent первого подэлемента, чей ограничивающий прямоугольник содержит точку, заданную в Event.Where. Для общих событий (т.е. не активных и не позиционированных) событие передается в HandleEvent каждого подэлемента группы в Z-порядке.

Примечание. Если поле EventMask подэлемента маскирует класс события, TGroup.HandleEvent ни-

когда не будет посылать события этого класса подэлементу. Например, по умолчанию EventMask из TView запрещает evMouseUp, evMouseMove и evMouseAuto, поэтому TGroup.HandleEvent никогда не будет посылать такие события стандартному TView.

См. также: FocusedEvents, PositionalEvents, константы событий evXXXX, TView.EventMask, методы HandleEvent

Insert procedure Insert(P: PView);

Вставляет видимый элемент, заданный P, в список подэлементов группы. Новый подэлемент помещается над всеми другими видимыми подэлементами. Если в подэлементе установлены флаги ofCenterX и/или ofCenterY, он центрируется в группе соответственно. Если видимый элемент имеет установленный флаг sfVisible, он будет показан в группе — иначе остается невидимым до тех пор, пока не будет показан специально. Если видимый элемент имеет установленным флаг ofSelectable, он становится текущим выбранным подэлементом.

InsertBefore procedure InsertBefore(P, Target: PView);

Вставляет видимый элемент, заданный P, перед видимым элементом, заданным Target. Если Target равен nil, видимый элемент размещается после всех видимых элементов группы.

См. также: TGrouup.Unsert, TGroup.Delete

Lock procedure Lock;

Блокирует группу, задерживая все записи, производимые подэлементами на экран, до тех пор, пока группа не будет разблокирована. Lock не имеет эффекта, если в группе нет кэш-буфера (см. ofBuffered и TGroup.Buffer). Lock работает, увеличивая счетчик блокировок, который соответственно уменьшается с помощью UnLock. Когда вызов UnLock уменьшает счетчик до 0, вся группа пишется на экран, используя образ, созданный в кэш-буфере.

Накапливая интенсивные операции прорисовки между вызовами Lock и UnLock, можно сократить или полностью избавиться от неприятного мерцания экрана. Например, TDeskTop.Tile и TDeskTop.Cascade используют Lock и UnLock для сокращения мерцания.

Примечание. Вызовы Lock и UnLock должны быть сбалансированы, иначе группа может остаться в постоянно заблокированном состоянии, что приведет к тому, что она не сможет вывести себя при необходимости.

См. также: TGroup.Unlock

PutSubViewPtr procedure PutSubViewPtr(var S: TStream; P: PView);

Сохраняет указатель подэлемента P в потоке S. PutSubViewPtr должна использоваться только внутри метода Store для записи значений указателей, которые позже будут читаться вызовами GetSubViewPtr в конструкторе Load.

См. также: TGroup.GetSubViewPtr, TGroup.Store, TGroup.Load

Redraw procedure Redraw;

Перерисовывает подэлементы группы в Z-порядке. TGroup.Redraw отличается от TGroup.Draw тем, что перерисовка никогда не производится выводом из кэш-буфера.

См. также: TGroup.Draw

SelectNext procedure SelectNext(Forwards: Boolean);

Если Forwards — True, SelectNext будет выбирать (делать текущим) следующий выбираемый подэлемент (подэлемент, в котором установлен бит ofSelectable) группы в Z-порядке. Если Forwards равен False, метод выбирает предыдущий выбираемый элемент.

См. также: константы флагов опций ofXXXX

SetData procedure SetData(var Rec); virtual;

Перекрывается: Редко

Перекрывает TView.SetData. Вызывает SetData для каждого подэлемента в порядке, обратном Z-порядку, увеличивая положение, заданное в Rec, на DataSize для каждого подэлемента.

См. также: TGroup.GetData, TView.SetData

SetState `procedure SetState(AState: Word; Enable: Boolean);virtual;`

Перекрывается: Редко

Перекрывает TView.SetState. Вначале вызывает унаследованный TView.State, затем обновляет подэлементы следующим образом: Если AState — sfActive, sfExposed или sfDragging, SetState вызывается для каждого подэлемента для его обновления. Если AState — sfFocused, то вызывается текущий выбранный подэлемент для своей активизации.

См. также: TView.SetState

Store `procedure Store(var S: TStream);`

Сохраняет всю группу в потоке, вначале вызывая унаследованный TView.Store, затем используя TStream.Put для вывода каждого подэлемента.

Если объектный тип, порожденный от TGroup, содержит поля, которые указывают на подэлементы, он должен использовать PutSubViewPtr внутри его Store для записи этих полей.

См. также: TView.Store, TGroup.PutSubViewPtr, TGroup.Load

Unlock `procedure Unlock;`

Разблокирует группу, уменьшая счетчик блокировки. Если счетчик блокировки становится 0, то вся группа выводится на экран, используя образ, созданный в кэш-буфере.

См. также: TGroup.Lock

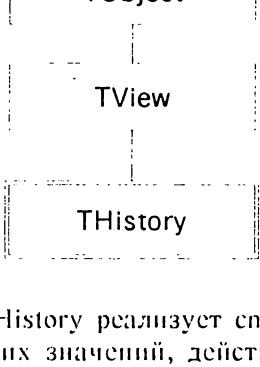
Valid `function Valid(Command: Word): Boolean; virtual;`

Перекрывает TView.Valid. Возвращает True, если вызовы Valid всех подэлементов возвращают True. TGroup.Valid используется в конце цикла обработки событий в TGroup.Execute для подтверждения,

что завершение разрешено. Модальное состояние не может быть завершено до тех пор, пока все вызовы Valid не вернут True. Подэлемент может вернуть False, если он хочет, чтобы управление осталось у него.

См. также: TView.Valid, TGroup.Execute

THistory — Dialogs



Объект THistory реализует список для отметки предыдущих значений, действий или выборов. Объекты THistory связываются с объектом TInputLine и со списком истории. Информация списка истории хранится в блоке памяти кучи. Когда блок заполняется, наиболее старые элементы истории удаляются, а новые добавляются.

Объект THistory показан как кнопка (↓) в конце строки ввода. Когда пользователь отмечает кнопку истории, Turbo Vision открывает окно истории (смотри THistoryWindow) с просмотром истории (смотри THistoryViewer), содержащем список предыдущих элементов.

Различные строки ввода могут использовать один список истории, используя одинаковый номер ID.

Поля

Link: PInputLine; — Только чтение

Указатель на связанный объект TInputLine.

HistoryID HistoryID: Word; Только чтение

Каждый список истории имеет уникальный номер ID, назначаемый программистом. Различные объекты историй в различных окнах могут разделять список истории, используя одинаковый ID номер.

Методы

Init constructor Init(var Bounds: TRect; ALink: PInputLine; AHistoryID: Word);

Создает объект THistory заданного размера, вызывая TView.Init, затем устанавливает поля Link и HistoryId значениями, заданными в аргументах. Поле Options устанавливается в ofPostProcess, а EventMask в evBroadcast.

См. также: TView.Init

Load constructor Load(var S: TStream);

Создает и инициализирует объект History из TStream, вызывая TView.Load(S) и читая Link и HistoryId из S.

См. также: TView.Store

Draw procedure Draw; virtual; Перекрывается: Редко

Рисует кнопку History в палитре по умолчанию.

GetPalette function GetPalette: PPalette; virtual; Перекрывается: Иногда

Возвращает указатель на умалчиваемую палитру CHistory.

Store procedure Store(var S:TStream);

Сохраняет объект THistory в TStream, вызывая TView.Store(S), затем записывает Link и HistoryId в S.

См. также: TView.Load

Палитра

Кнопки истории используют палитру по умолчанию CHistory для отображения 14 и 20-го элементов в палитру стандартного диалогового окна.

THistoryViewer Dialogs

THistoryViewer — прямой потомок TListViewer. Он используется системой списка истории и появляется внутри окна истории при отметке кнопки истории. Для детального описания взаимодействия THistory, THistoryWindow и THistoryViewer смотри THistory в этой главе.

Поля

HistoryID HistoryID: Word; Только чтение

HistoryId — это ID номер списка истории, отображаемого в этом видимом элементе.

Методы

Init constructor Init(var Bounds:TRect; AHScrollBar, AVScrollBar: PScrollBar; AHistoryID: Word);

Init инициализирует видимый элемент просмотра списка, вначале вызывая TListViewer.Init для установки границ, одной колонки и двух полос скроллинга, передаваемых в AHScrollBar и AVScrollBar. Видимый элемент связывается затем со списком истории, с полем HistoryId, установленным в значение, переданное в AHistory. Этот список проверяется затем на длину так, что диапазон списка устанавливается в число списка. Первый элемент в списке истории дан как активный. Диапазон горизонтального скроллинга устанавливается в соответствии с самым широким элементом списка.

См. также: TListViewer.Init

GetPalette function GetPalette: PPalette; virtual;

Перекрывается: Иногда

Возвращает указатель на умалчиваемую палитру CHistoryViewer.

GetText function GetText(Item: Integer; MaxLen: Integer): String; virtual;

Перекрывается: Редко

Возвращает строку Item в связанном списке истории. GetText вызывается виртуальным методом Draw для каждого видимого элемента в списке.

См. также: TListViewer.Draw, HistoryStr function

HandleEvent procedure HandleEvent(var Event: TEvent); virtual;

Перекрывается: Иногда

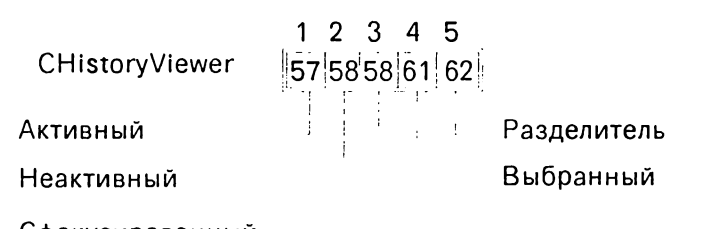
Видимый элемент просмотра истории управляет двумя видами событий; все другие передаются в TListViewer.HandleEvent. Двойная отметка или нажатие клавиши Enter будут завершать модальное состояние окна истории с командой cmOK. При нажатии клавиши Esc или получении командного события cmCancel выбор списка истории будет отменен.

См. также: TListViewer.HandleEvent

HistoryWidth function HistoryWidth: Integer;

Возвращает длину самой длинной строки в списке истории, связанном с HistoryId.

Палитра



Объекты просмотра истории используют палитру по умолчанию CHistoryViewer для отображения в 6 и 7 элементы палитры стандартного диалогового окна.

THistoryWindow — Dialogs

THistoryWindow — это специализированный наследник от TWindow, используемый для объекта просмотра списка истории, когда пользователь отмечает кнопку истории, стоящую за строкой ввода. По умолчанию окно не имеет заголовка и номера. Рамка окна истории имеет закрывающую кнопку, поэтому окно может быть закрыто, но не может изменить размер или масштабироваться.

Для деталей по использованию списков истории и связанных с ними объектов см. THistory в этой главе.

Поля

Viewer Viewer: PListViewer;

Viewer указывает на список просмотра окна истории.

Методы

Init constructor Init(var Bounds: TRect; HistoryId: Word);

Вызывает TWindow.Init для установки окна с заданными границами, пустой строкой заголовка и без номера окна (wnNoNumber). Поле TWindow.Flags устанавливается в wfClose, чтобы обеспечить закрывающую кнопку, и объект просмотра истории создается, чтобы показать элементы списка истории, заданные через HistoryID.

См. также: TWindow.Init, THistoryWindow.InitViewer

GetPalette function GetPalette: PPalette; virtual;

Перекрывается: Иногда

Возвращает указатель на палитру по умолчанию CHistoryWindow.

GetSelection function GetSelection: String; virtual;

Перекрывается: Никогда

Возвращает строковое значение активного элемента из просмотра истории.

См. также: THistoryViewer.GetText

InitViewer procedure InitViewer(HistoryOd: Word); virtual;

Перекрывается: Никогда

Создает и вставляет объект THistoryViewer внутри границ окна истории со списком, заданным через HistoryId. Стандартные полосы скроллинга размещены на рамке окна для скольжения по списку.

См. также: THistoryViewer.Init

Палитра

Объекты окна истории используют по умолчанию палитру CHistoryWindow для отображения на элементы с 19 по 25-й в палитре стандартного диалогового окна.

	1	2	3	4	5	6	7
CHistoryWindow	10	11	12	13	14	14	14

Пассивная рамка

Активная рамка

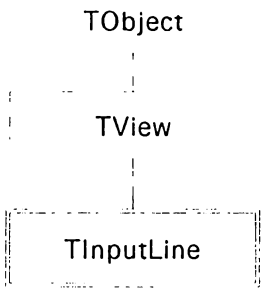
Кнопка рамки

Выбранный текст

Нормальный текст

Элементы управления

Область страницы



Объект TInputLine обеспечивает редактор строк ввода. Он управляет вводом с клавиатуры и мышки и перемещением помеченных блоков в различных функциях редактирования строки (см. TInputLine.HandleEvent). Выбранный текст удаляется и заменяется первым введенным текстом. Если MaxLen больше размера по X (Size.X), поддерживается горизонтальный скроллинг, который указывается правой и левой стрелками.

Методы GetData и SetData предназначены для записи и чтения строк данных (через поле указателя Data) в данную запись. TInputLine.SetState упрощает перерисовку видимого элемента соответствующим цветом, когда состояние изменяется из или в sfActive и sfSelected.

Строка ввода часто имеет ассоциированные с ним объекты TLabel и/или THistory.

TInputLine можно расширить для обработки типов данных отличных от строк. Чтобы сделать это, вы добавляете новые поля и перекрываете методы Init, Load, Store, Valid, DataSize, GetData и SetData. Например, чтобы определить строку ввода числа, вы можете задать минимальные и максимальные допустимые значения, которые будут проверяться функцией Valid. Эти минимальные и максимальные поля будут загружаться и сохраняться в потоке

методами Load и Store. Valid модифицируется для проверки того, что значение находится в допустимом диапазоне. DataSize модифицируется для включения размера диапазона новых полей (вероятно, SizeOf(Longint) для каждого). В этом примере не обязательно добавлять поле для хранения числового значения. Оно может храниться как строковое значение (которое уже обрабатывается в TInputLine) и преобразовываться из строки в числовое значение и обратно методами GetData и SetData соответственно.

Поля

Data Data: PString; Чтение/Запись

Указатель на строку, содержащую редактируемую информацию.

MaxLen MaxLen: Integer; Только чтение

Максимальная длина, допустимая для строки, включая байт длины.

См. также: TInputLine.DataSize

CurPos CurPos: Integer; Чтение/Запись

Индекс на точку вставки (т.е. на текущую позицию курсора).

См. также: TInputLine.SelectAll

FirstPos FirstPos: Integer; Чтение/Запись

Индекс на первый отображаемый символ.

См. также: TInputLine.SelectAll

SelStart SelStat: Integer; Только чтение

Индекс на начало выбранной области (т.е. на первый символ отмеченного блока).

См. также: TInputLine.SelectAll

SelEnd SelEnd: Integer; Только чтение

Индекс на конец выбранной области (т.е. на последний символ отмеченного блока).

См. также: TInputLine.SelectAll

Init constructor Init(var Bounds: TRect; AMaxLen: Integer);

Создает прямоугольник ввода с заданными значениями, вызывая TInputLine.Init. State устанавливается в sfCursorVis, Options устанавливается в (ofSelectable+ofFirstClick), и MaxLen устанавливается в AMaxLen. Память распределяется и очищается под AMaxLen + 1 байт и поле Data устанавливается для указания этого распределения.

См. также: TView.Init, TView.sfCursorVis, TView.ofSelectable, TView.ofFirstClick

Load constructor Load(var S: TStream);

Создаёт и инициализирует объект TInputLine, вызывая TView. Load(S) для загрузки видимого элемента из потока, затем читает целочисленные поля, используя S.Read, распределяет MaxLen+1 байт через Data, вызывая GetMem. Наконец, устанавливает байт длины строки и загружает данные из потока двумя дополнительными вызовами S.Read. Load используется совместно с TInputLine.Store для сохранения и восстановления объектов TInputLine из TStream. Перекрывайте этот метод, если вы определили потомков, содержащих дополнительные поля.

См. также: TView.Load, TInputLine.Store, TStream.Read

Done destructor Done; virtual; Перекрывается: Редко

Освобождает память Data, затем вызывает TView.Done для разрушения объекта TInputLine.

См. также: TView.Done

DataSize function DataSize: Word; virtual;

Перекрывается: Иногда

Возвращает размер записи для TInputLine.GetData и TInputLine.SetData. По умолчанию возвращается MaxLen+1. Перекройте этот метод, если вы определили потомков для обработки других типов данных.

См. также: TInputLine.GetData,
TInputLine.SetData

Draw procedure Draw; virtual; Перекрывается: Редко
Рисует прямоугольник ввода и его данные. Прямоугольник рисуется соответствующим цветом в зависимости от того, является ли он sfFocused или нет (т.е. находится ли в видимом элементе курсор или нет), и стрелки рисуются, если строка ввода превышает размер видимого элемента (в любом или в обоих направлениях). Любые выбранные (отмеченный блок) символы рисуются соответствующим цветом.

GetData procedure GetData(var Rec); virtual;
Перекрывается: Иногда
Записывает DataSize байт из строки Data^ в запись. Используется с TInputLine.SetData в ряде программ, например во временной памяти или при передаче строки ввода в другие видимые элементы. Перекройте этот метод, если вы определяете потомка для обработки нестроковых типов данных. Используйте этот метод для преобразования ваших типов данных в строку, редактируемую с помощью TInputLine.
См. также: TInputLine.DataSize,
TInputLine.SetData

GetPalette function GetPalette: PPalette; virtual;
Перекрывается: Иногда
Возвращает указатель на палитру по умолчанию CInputLine.

HandleEvent procedure HandleEvent(var Event: TEvent);
virtual;
Перекрывается: Иногда
Вызывает TView.HandleEvent, которая обрабатывает все события от мышки и клавиатуры, если прямоугольник ввода выбран. Этот метод реализует стандартные функции редактирования.
Функции редактирования включают: отметку блока мышкой; удаление блока, вставку или перекры-

тие элемента управления с автоматическим изменением формы курсора; автоматический и ручной скроллинг (зависит от относительных размеров строки Data и Size.X); ручной горизонтальный скроллинг через отметку мышкой на кнопках стрелок; ручное движение курсора с помощью стрелок Home и End; удаление символа и блока с помощью Del и Ctrl-G. Видимый элемент перерисовывается при необходимости и поля TInputLine соответственно изменяются.

См. также: sfCursorIns, TView.HandleEvent, TInputLine.SelectAll

SelectAll procedure SelectAll(Enable: Boolean);

Устанавливает CurPos, FirstPos и SelStart в 0. Если Enable установлен в True, SelEnd устанавливается в Length(Data^), выбирая таким образом всю строку ввода. Если Enable установлен в False, SelEnd устанавливается в 0 снимая таким образом выбор всей строки. Наконец, видимый элемент перерисовывается вызовом DrawView.

См. также: TView.DrawView

SetData procedure SetData(var Rec); virtual;

Перекрывается: Иногда

По умолчанию читает DataSize байт из записи в строку Data^ и вызывает SelectAll(True) для установки CurPos, FirstPos и SelStart в 0; SelEnd устанавливается на последний символ Data^ и видимый элемент перерисовывается. Перекройте этот метод, если вы определили потомка для обработки нестроковых типов данных. Используйте этот метод для преобразования из строки в ваш тип данных после редактирования с помощью InputLine.

См. также: TInputLine.DataSize, TInputLine.GetData, TView.DrawView

SetState procedure AetState(AState: Word; Enable: Boolean); virtual;

Перекрывается: Редко

Вызывается, когда прямоугольник ввода должен быть перерисован (например при изменении палитры) после изменения State. Вызов TView.SetState устанавливает или очищает поле State видимого элемента заданными битами AState. Затем, если AState — sfSelected или если AState — sfActive и прямоугольник ввода — sfSelected, то вызывается SelectAll(Enable).

См. также: TView.SetState, TView.DrawView

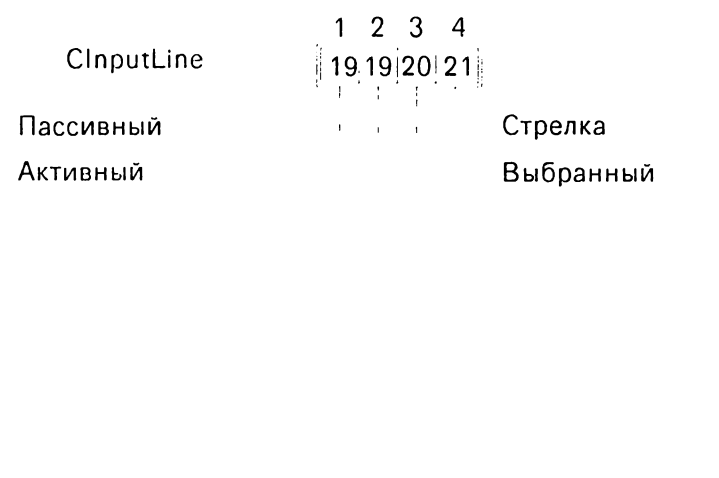
Store `procedure Store(var S: TStream);`

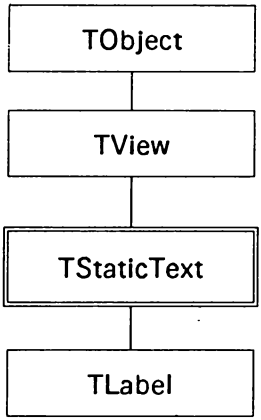
Сохраняет видимый элемент в потоке, вызывая TView.Store(S), затем сохраняет 5 целочисленных полей и строку Data вызовами S.Write. Используется совместно с TInputLine.Load для сохранения и чтения всего объекта TInputLine. Перекройте этот метод, если вы определяете потомка, который содержит дополнительные поля.

См. также: TView.Store, TInputLine.Load, TStream.Write

Палитра

Строки ввода используют палитру по умолчанию CInputLine для отображения в элементы с 19 по 21-й в палитре стандартного диалогового окна.





Объект TLabel — это текст в видимом элементе, который может быть выбран (подсвечен) отметкой мышки, клавишами курсора или коротким выбором "Alt-буква". Метка обычно присоединена через указатель PView к другому видимому элементу управления, такому, как строка ввода, кластер или просмотр списка для пояснения пользователю. Выбор (или нажатие) метки будет выбирать присоединенный элемент управления. Метка также будет подсвечиваться, когда выбран связанный элемент управления.

Поля

Link Link: PView; Только чтение
Указывает на элемент управления, связанный с этой меткой.

Light Light: Boolean; Только чтение
Если True — метка, а связанный с ней элемент управления будут выбираться и подсвечиваться.

Методы

Init constructor Init(var Bounds: TRect; AText: String; ALink: PView);

Создает объект TLabel заданного размера, вызывая TStaticText.Init, затем устанавливает поле Link в ALink для связывания с элементом управления (задайте Alink равным nil, если элемент управления не требуется). Поле Option устанавливается в ofPreProcess и ofPostProcess. EventMask устанавливается evBroadcast. Поле AText назначается полю Text через TStaticText.Init. AText может задать клавишу короткого набора для метки, окружив соответствующую букву "~".
См. также: TStaticText.Init

Load constructor Load(var S: TStream);
Создает и загружает объект TLabel из потока, вызывая TStaticText.Load, GetPeerViewPtr(S, Link) для установки связи с ассоциированным элементом управления (если он есть).
См. также: TLabel.Store

Draw procedure Draw; virtual; Перекрывается: Никогда
Рисует видимый элемент соответствующим цветом из палитры по умолчанию.

GetPalette function GetPalette: PPalette; virtual;
Перекрывается: Иногда
Возвращает указатель на палитру по умолчанию CLabel.

HandleEvent procedure HandleEvent(var Event: TEvent); virtual;
Перекрывается: Никогда
Обрабатывает все события, вызывая TStaticText.HandleEvent. Если evMouseDown или если получено событие от клавиши короткого набора, выбирается связанный элемент управления (если он есть). Этот метод также обрабатывает общие события cmReceivedFocus и cmReleasedFocus от связанного элемента управления для настройки значения поля Light и перерисовки метки.
См. также: TView.HandleEvent, cmXXXX константы команд

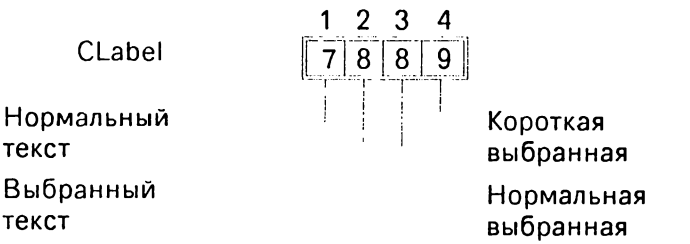
Store procedure Store(var S: TStream);

Сохраняет видимый элемент в потоке, вызывая TStaticText.Store, затем записывает связь с ассоциированным элементом управления, вызывая PutPeerViewPtr.

См. также: TLabel.Load

Палитра

Метки используют палитру по умолчанию CLabel для отображения в элементы 7, 8 и 9 палитры стандартного диалога.



TListBox — Dialogs

TObject

TView

TListViewer

TListBox

TListBox, порожденный от TListViewer, помогает вам создавать наиболее часто используемые окна списков, отображающих коллекции строк, таких,

как имена файлов. Объекты TListBox отображают списки таких элементов в одну или более колонок с возможной вертикальной полосой скроллинга. Горизонтальные полосы скроллинга не поддерживаются в TListViewer. Наследуемые методы TListViewer позволяют вам выбрать (и подсветить) элементы мышкой или через клавиатуру. TListBox не перекрывает TListViewer.HandleEvent и TListViewer.Draw, поэтому вы должны просмотреть их описание до использования TListBox в своих программах.

TListBox имеет дополнительное поле List, которое указывает на объект TCollection, содержащий выводимые и выбираемые элементы. Ответственность за вставку данных в TCollection лежит на вас так же, как и действия, выполняемые при выборе элемента.

TListViewer наследует метод Done от TView, поэтому вы также отвечаете за освобождение содержимого List при окончании работы. Вызов NewList будет освобождать старый список, поэтому вызов NewList(nil) и последующее освобождение окна списка будут освобождать все.

Поля

List List: PCollection; Только чтение

List указывает на коллекцию элементов для просмотра. Это может быть коллекция PString, представляющая текстовые элементы.

Методы

Init constructor Init(var Bounds: TRect; ANumCols: Word; AScrollBar: PScrollBar);

Создает окно списка с заданным размером, числом колонок и вертикальной полосой скроллинга, указываемой указателем AScrollBar. Этот метод вызывает TListViewer.Init с аргументом горизонтальной полосы скроллинга nil.

Поле List первоначально nil (пустой список) и наследуемое поле Range устанавливается в 0. Ваша

программа должна задать TCollection, содержащую строки (или другие объекты для вывода). Поле List должно быть установлено на эту коллекцию с использованием NewList.

См. также: TListViewer.Init, TListBox.NewList

Load constructor Load(var S: TStream);

Создает объект TListBox и загружает его значениями из TStream. Этот метод вызывает метод TListViewer.Load, затем устанавливает List, читая указатель List из S с помощью S.Get.

См. также: TListViewer.Load, TListBox.Store, TStream.Get

DataSize function DataSize: Word; virtual;

Перекрывается: Иногда

Возвращает размер читаемых и записываемых данных для записей, передаваемых в TListBox.GetData и TListBox.SetData. Эти три метода полезны для инициализации групп. По умолчанию TListBox.DataSize возвращает размер указателя плюс размер слова (для List и выбранного элемента). Вам может потребоваться перекрыть этот метод для вашей программы.

См. также: TListBox.GetData, TListBox.SetData

GetData procedure GetData(var Rec); virtual;

Перекрывается: Иногда

Записывает данные объекта TListBox в запись. По умолчанию этот метод пишет в Rec текущие поля List и Focused. Вам может потребоваться перекрыть этот метод для вашей программы.

См. также: TListBox.DataSize, TListBox.SetData

GetText function GetText(Item: Integer; MaxLen: Integer): String; virtual;

Перекрывается: Иногда

Возвращает строку из вызываемого объекта TListBox. По умолчанию возвращаемая строка получается из элемента Item в TCollection, используя PString(List^.At(Item))^. Если List содержит не строковые объекты, вам необходимо перекрыть

этот метод. Если List — nil, GetText возвращает пустую строку.
См. также: TCollection.At

NewList procedure NewList(AList: PCollection); virtual;

Перекрывается: Редко

Если AList не nil, новый список, заданные в AList, заменяет текущий List. Наследуемое поле Range устанавливается в поле Count новой TCollection и первый элемент активизируется вызовом FocusItem(0). Наконец, новый список отображается вызовом DrawView. Заметим, что если предыдущее поле List не nil, оно освобождается до назначения нового списка значений.
См. также: TListBox.SetData, TListViewer.SetRange, TListViewer.FocusItem, TView.DrawView

SetData procedure SetData(var Rec); virtual;

Перекрывается: Иногда

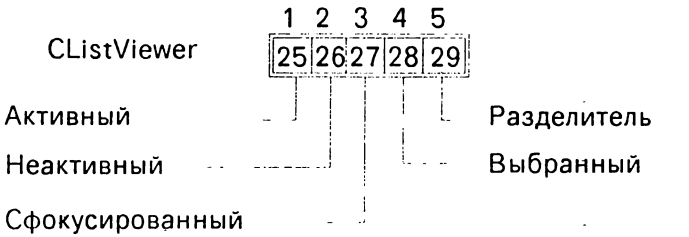
Заменяет текущий список со значениями List и Focused, считанными из Rec. SetData вызывает NewList так, чтобы новый список отображался с корректным активным элементом. Как с GetData и DataSize, вам может потребоваться перекрыть этот метод для вашей программы.
См. также: TListBox.DataSize, TListBox.GetData, TListBox.NewList

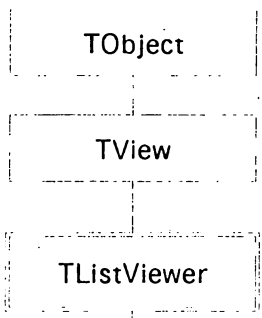
Store procedure Store(var S:TStream);

Записывает окно списка в TStream, вызывая TListView.Store, затем выводит коллекцию в список, вызывая S.Put(List).
См. также: TListBox.Load, TListViewer.Store, TStream.Put

Палитра

Окна списка используют палитру по умолчанию CListViewer, отображая элементы с 26 по 29 в палитру стандартной программы.





TListBox

Тип объекта TListViewer — это базовый тип, из которого порождаются просмотры списков различного вида такие, как TListBox. Основные поля и методы TListViewer предоставляют следующие функции:

- Видимый элемент для отображения связанных списков элементов (но не списков).
- Управление одним или двумя полосами скроллинга.
- Скроллинг списков по двум координатам.
- Загрузка и сохранение видимого элемента и его полос скроллинга из и в TStream.
- Возможность выбора элементов списка мышкой или клавишами.

- Метод Draw поддерживающий изменение размера и скроллинг.

TListViewer имеет абстрактный метод GetText так, что вам потребуется предоставить механизм создания и манипуляции отображаемых элементов текста. TListViewer не имеет собственного механизма запоминания списка. Используйте его для отображения скроллингуемых списков, массивов, связанных списков или подобных структур данных. Вы также можете использовать его наследников таких, как TListBox, которые ассоциируют коллекцию с просмотром списка.

Поля

HScrollBar HScrollBar: PScrollBar; Только чтение

Указатель на горизонтальную полосу скроллинга, связанную с этим видимым элементом. Если nil, видимый элемент не имеет такой полосы скроллинга.

VScrollBar VScrollBar: PScrollBar; Только чтение

Указатель на вертикальную полосу скроллинга, связанную с этим видимым элементом. Если nil, видимый элемент не имеет такой полосы скроллинга.

NumCols NumCols: Integer; Тодько чтение

Число колонок в элементе управления списком.

TopItem TopItem: Integer; Чтение/Запись

Номер верхнего отображаемого элемента. Элементы нумеруются от 0 до Range–1. Это число зависит от числа колонок, размера видимого элемента и значения Range.

См. также: Range

Focused Focused: Integer; Только чтение

Номер активного элемента. Элементы нумеруются от 0 до Range–1. Первоначально устанавливается в 0 — первый элемент, Focused может изменяться отметкой мышки или выбором через пробел.

См. также: Range

Range Range: Integer; Только чтение

Общее число элементов в списке. Элементы нумеруются от 0 до Range–1.

TListViewer.SetRange

Методы

Init constructor Init(var Bounds: TRect; ANumCols: Integer; AHScrollBar, AVScrollBar: PScrollBar);

Создает и инициализирует объект TListViewer заданного размера, вызывая TView.Init. Поле NumCols устанавливается в ANumCols. Options устанавливается в (ofFirstClick + ofFirstSelectable) так, что выбор мышкой этого элемента будет передаваться в TListViewer.HandleEvent. EventMask устанавливается в evBroadcast. Начальные значения Range и Focused — 0. Указатели на вертикальную и/или горизонтальную полосы скроллинга можно задать через аргументы AVScrollBar и AHScrollBar. Если вам не нужны полосы скроллинга, можно установить один из них или оба в nil. Эти значения аргументов назначаются полям VScrollBar и HScrollBar.

Если вы задали полосы скроллинга, их поля PgStep и ArStep будут настраиваться в соответствии с размером TListViewer и числом колонок. Например, для одноколоночного TListViewer вертикальный PgStep равен Size.Y–1, а вертикальный ArStep равен 1.

См. также: TView.Init, TScrollBar.SetStep

Load Load constructor Load(var S: TStream);

Создает объект TListViewer, вызывая TView.Load. Полосы скроллинга, если они есть, также загружаются из потока с использованием вызовов GetPeerViewPtr. Все целочисленные поля загружаются с использованием S.Read.

См. также: TView.Load, TListViewer.Store

ChangeBounds ChangeBounds(var Bounds: TRect); virtual;

Перекрывается: Никогда

Изменяет размер объекта TListViewer, вызывая TView.ChangeBounds. Если назначена горизонтальная полоса скроллинга, этот метод при необходимости настраивает PgStep.
См. также: TView.ChangeBounds, TScrollBar.ChangeStep

Draw procedure Draw; virtual; Перекрывается: Никогда
Рисует объект TListViewer палитрой по умолчанию, вызывая GetText для каждого отображаемого элемента, принимая во внимание активные и выбранные элементы и является ли видимый элемент sfActive.
См. также: TListViewer.GetText

FocusItem procedure FocusItem(Item: Integer); virtual;
Перекрывается: Никогда
Делает данный элемент активным, устанавливая поле Focused в Item. Этот метод также устанавливает поле Value вертикальной полосы скроллинга (если есть) в Item и настраивает поле TopItem.
См. также: TListViewer.IsSelected, TScrollBar.SetValue

GetPalette function GetPalette: PPalette; virtual;
Перекрывается: Иногда
Возвращает указатель на палитру по умолчанию.

GetText function GetText(Item: Integer; MaxLen: Integer): String; virtual;
Перекрывается: Всегда
Это абстрактный метод. Порожденные типы должны определить механизм для возвращения строки, не превышающей MaxLen по индексу, заданному в Item.
См. также: TListViewer.Draw

IsSelected function IsSelected(Item: Integer): Boolean; virtual;
Перекрывается: Никогда
Возвращает True, если Item активный, т.е. если Item = Focused.

См. также: TListViewer.FocusItem

HandleEvent procedure HandleEvent(var Event: TEvent); virtual;

Перекрывается: Редко

Обрабатывает события, вызывая TView.HandleEvent. Отметки мышкой и «автоматические» движения по списку будут изменять активный элемент. Активные элементы могут быть выбраны двойным нажатием мышки. Обрабатываются события от клавиатуры: пробел выбирает текущий активный элемент; клавиши стрелок, PgUp, PgDn, Ctrl-PgUp, Ctrl-PgDn, Home и End изменяют активный элемент. Наконец, общие сообщения от полос скроллинга обрабатываются, изменяя активный элемент и перерисовывая видимый элемент.

См. также: TView.HandleEvent, TListViewer.FocusItem

SelectItem procedure SelectItem(Item: Integer); virtual;

Перекрывается: Иногда

Абстрактный метод для выбора элемента, индексируемого через Item.

См. также: TListViewer.FocusItem

SetRange procedure SetRange(ARange: Integer);

Устанавливает поле Range в ARange. Если вертикальная полоса скроллинга была задана, ее параметры настраиваются. Если текущий активный элемент выходит за новый Range, поле Focused устанавливается в 0.

См. также: TListViewer.Range, TScrollBar.SetParams

SetState procedure SetState(AState: Word; Enable: Boolean); virtual;

Перекрывается: Редко

Вызывает TView.SetState для изменения состояния объекта TListViewer, если Enable — True. В зависимости от аргумента AState это приводит к отображению или скрытию видимого элемента. Дополни-

тельно, если AState — sfSelected и sfActive, полосы скроллинга перерисовываются; если AState — sfSelected, но не sfActive, полосы скроллинга скрываются.

См. также: TView.SetState, TScrollBar.Show, TScrollBar.Hide

Store `procedure Store(var S: TStream);`

Вызывает TView.Store для сохранения объекта TListViewer в потоке, затем сохраняет объекты полос скроллинга (если они есть) используя PutPeerViewPtr, наконец сохраняет целочисленные поля через S.Write.

См. также: TView.Store, TListViewer.Load

Палитра

Списки просмотра используют палитру по умолчанию CListViewer, отображая элементы с 26 по 29-й в палитру стандартной программы.

	1	2	3	4	5
CListViewer	25	26	27	28	29

Активный

Неактивный

Сфокусированный

Разделитель

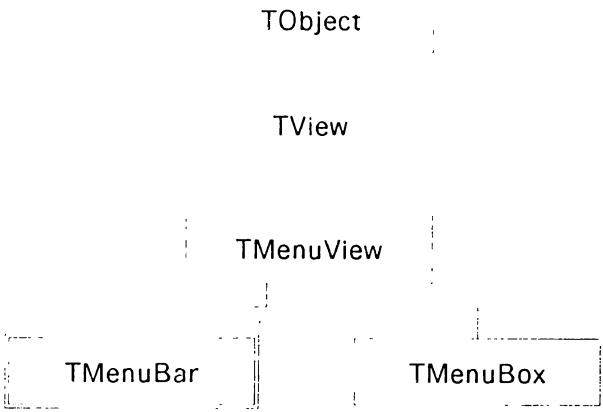
Выбранный

TMenuBar — Menus

Объекты TMenuBar представляют полосы горизонтального меню, из которого меню может быть выбрано через:

- прямую отметку;
- F10 и короткую клавишу;
- выбор (подсветку) и нажатие Enter;
- горячие клавиши.

Выборы главного меню отображаются в верхней полосе меню. Они представлены объектом типа



TMenuBar обычно принадлежащему объекту TApplication. Подменю отображаются в объектах типа TMenuBox. TMenuBar и TMenuBox порождаются от абстрактного типа TMenuView.

Для большинства программ на Turbo Vision вы не включаете прямо объекты меню. Перекрывая TApplication.InitMenuBar соответствующим набором вложенных вызовов New, NewSubMenu, NewItem и NewLine, Turbo Vision выполняет это.

Методы

Init constructor Init(var Bounds: TRect; AMenu: PMenu);
Создает полосу меню, заданную через Bounds, вызывая TMenuView.Init. GrowMode установлена в gfGrouHiX. Поле Options установлено в ofPreprocess для возможности работы с горячими клавишами. Поле меню установлено в AMenu, задавая элементы меню.
См. также: TMenuView.Init, gfXXXX флаги grow mode, ofXXXX флаги опций, TMenuView.Menu

Draw procedure Draw; virtual; Перекрывается: Редко
Рисует полосу меню палитрой по умолчанию. Поля Name и Disabled каждой записи TMenuItem в связанном списке читаются, чтобы установить эле-

менты меню в корректный цвет. Текущий элемент Current подсвечивается.

GetItemRect procedure GetItemRect(Item: PMenuItem;
var R: TRect); virtual;

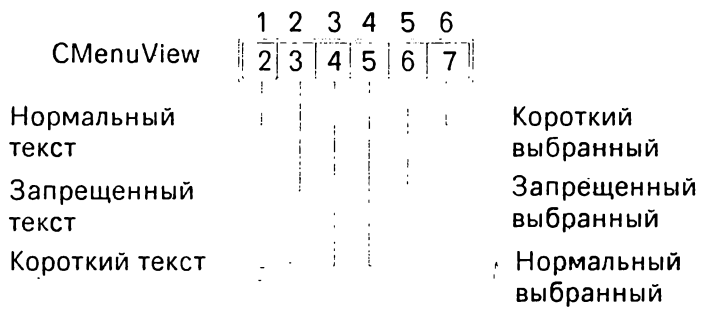
Перекрывается: Никогда

Перекрывает абстрактный метод в TMenuView. Возвращает прямоугольник, занимаемый элементом меню в R. Используется для определения, находится ли отметка мышки в данном элементе меню.

См. также: TMenuView.GetItemRect

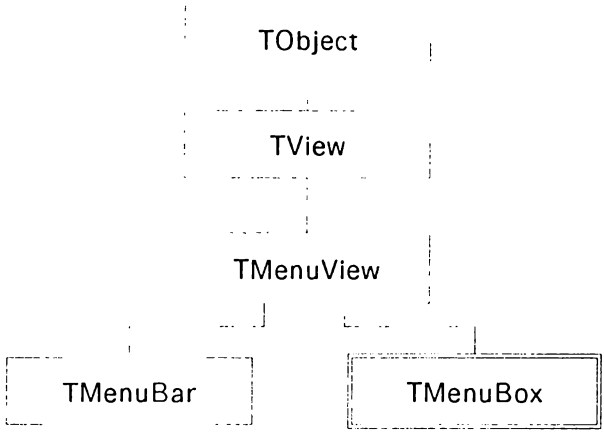
Палитра

Полосы меню, как и все видимые элементы меню, используют палитру по умолчанию CMenuView для отображения элементов со 2 по 7 в палитру стандартной программы.



TMenuBox — Menus

Объекты MenuBox представляют вертикальные прямоугольники меню. Они могут содержать произвольный список выбираемых действий, включая элементы подменю. Как и в полосе меню, для указания запрещенных элементов используется цвет. Прямоугольники меню могут создаваться как подменю полосы меню или других прямоугольников меню или могут использоваться как отдельные выпадающие меню.



Методы

Init constructor Init(var Bounds: TRect; Amenu: PMenu; AParentMenu: PMenuView);

Init настраивает параметр Bounds в соответствии с шириной и длиной элементов в AMenu, затем создает прямоугольник меню, вызывая TMenuView.Init.

Бит ofPreProcess в поле Options установлен так, чтобы могли использоваться горячие клавиши. Установка State включает sfShadow. Поле меню установлено в AMenu, которое задает элементы меню. Поле ParentMenu установлено в AParentMenu.

См. также: TMenuView.Init, sfXXXX флаги состояний, ofXXXX флаги опций, TMenuView.Menu, TMenuView.ParentMenu

Draw procedure Draw; virtual; Перекрывается: Редко

Рисует рамку прямоугольника меню и элементы меню цветом по умолчанию.

GetItemRect procedure GetItemRect(Item: PMenuItem; var R: TRect; virtual;

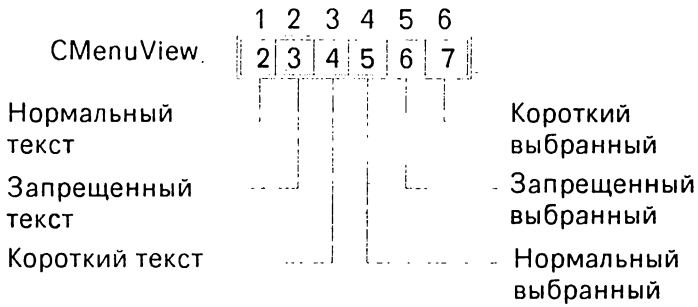
Перекрывается: Редко

Перекрывает абстрактный метод в TMenuView. Возвращает прямоугольник, занятый элементом

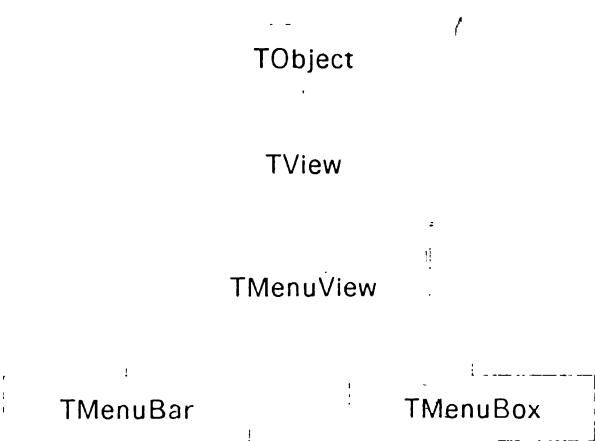
меню. Используется для определения, находится ли отметка мышкой на данном элементе меню.
См. также: TMenuView.GetItemRect

Палитра

Прямоугольники меню, как и все видимые элементы меню, используют палитру по умолчанию CMenuView для отображения элементов со 2 по 7 в палитру стандартной программы.



TMenuView — Menus



TMenuView предоставляет тип абстрактного меню, из которого порождаются полосы и прямоугольники меню. Вы, вероятно, никогда не будете создавать экземпляров TMenuView.

Поля

ParentMenu ParenMenu: PMenuView; Только чтение

Указатель на объект TMenuView (или порожденный от него), который владеет этим меню. Заметим, что TMenuView — не группа. Здесь принадлежность намного проще, чем для TGroup, позволяя вложенность меню: выбор подменю и обратный возврат в «родительское» меню. Выборы из полос меню, например, обычно приводят к «выпаданию» подменю. В этом случае полоса меню — предок прямоугольника меню.

См. также: TMenuBox.Init

Menu Menu: PMenu; Только чтение

Указатель на запись TMenu для этого меню, которая содержит связанный список элементов меню. Указатель меню позволяет обращаться ко всем полям элементов меню в видимом элементе меню.

См. также: TMenuView.FindItem, TMenuView.GetItemRect, тип TMenu

Current Current: PMenuItem; Только чтение

Указатель на текущий выбранный элемент меню.

Методы

Init constructor Init(var Bounds: TRect);

Вызывает TView.Init, чтобы создать объект TMenuView размера Bounds. По умолчанию EventMask установлено в evBroadcast. Этот метод не предназначен для использования с экземплярами объектов TMenuView. Он предназначен для вызова из порожденных типов TMenuBar и TMenuBox.

См. также: TView.Init, evBroadcast, TMenuBar.Init, TMenuBox.Init

Load constructor TMenuView.Load(var S: TStream);

Создает объект TMenuView и загружает его из потока S вызывая TView.Load, а затем загружая элементы в список меню.

См. также: TView.Load, TMenuView.Store

Execute function Execute: Word; virtual;

Перекрывается: Никогда

Выполняет видимый элемент меню до тех пор, пока пользователь не выберет элемент меню или не отменит этот процесс. Возвращает команду, назначенную выбранному элементу меню, или 0, если меню было отменено. Этот метод должен вызываться только из ExecView.

См. также: TGroup.ExecView

FindItem function FindItem(Ch: Char): PMenuItem;

Возвращает указатель на элемент меню, который имеет Ch как горячую клавишу (подсвеченный символ). Возвращает nil, если такой элемент не найден или этот элемент запрещен. Заметим, что для Ch не различаются прописные и строчные буквы.

GetItemRect procedure GetItemRect(Item: PMenuItem; var R: TRect); virtual;

Перекрывается: Всегда

Этот метод возвращает в R прямоугольник, занимаемый данным элементом меню. Используется для определения, не был ли отмечен данный элемент мышкой. Наследники TMenuView должны перекрывать этот метод для того, чтобы откликаться на события от мышки.

См. также: TMenuBar.GetItemRect, TMenuBox.GetItemRect

GetHelpCtx function GetHelpCtx: Word; virtual;

Перекрывается: Иногда

По умолчанию этот метод возвращает контекст подсказки текущего элемента меню. Если это hcNoContext, выбирается текущий контекст роди-

тельского меню. Если родительского меню нет, GetHelpCtx возвращает hcNoContext.
См. также: hcXXXX константы контекста help

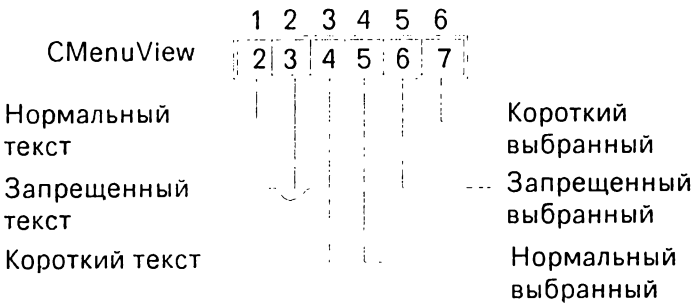
GetPalette function GetPalette: PPalette; virtual;
Перекрывается: Иногда
Возвращает указатель на палитру по умолчанию CMenuView.

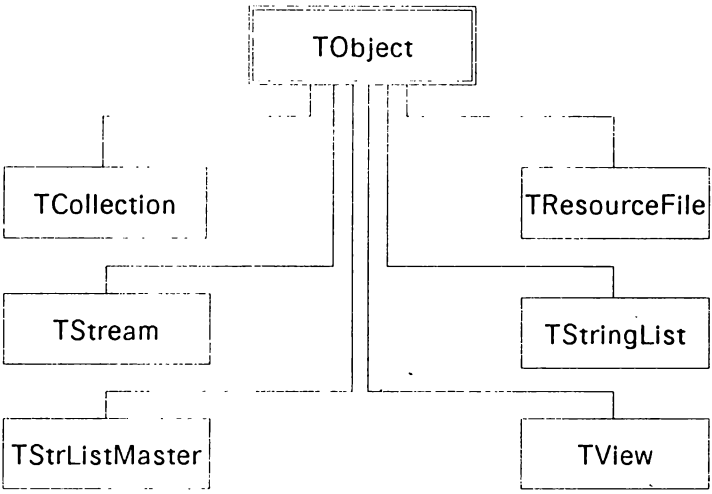
HandleEvent procedure HandleEvent(var Event: TEvent); virtual;
Перекрывается: Никогда
Вызывается для обработки событий меню. Определяет, какой элемент меню был выбран мышкой или клавиатурой (включая горячие клавиши), и генерирует соответствующее командное событие через PutEvent.
См. также: TView.HandleEvent, TView.PutEvent

HotKey function HotKey(KeyCode: Word): PMenuItem;
Возвращает указатель на элемент меню, связанный с горячей клавишей, заданной через KeyCode. Возвращает nil, если не существует такого элемента меню или если элемент запрещен. Горячие клавиши — это обычно функциональные клавиши или комбинации Alt ключей и определяются аргументами в вызовах NewItem и NewSubMenu в InitMenuBar. Этот метод используется в TMenuView.HandleEvent для определения, будет ли событие от клавиатуры выбирать элемент меню.

Store procedure Store(var S: TStream);
Сохраняет объект TMenuView (и его подменю) в потоке S, вызывая TView.Store, а затем записывая каждый элемент меню в поток.
См. также: TMenuView.Load

Все видимые элементы меню используют палитру по умолчанию CMenuView для отображения элементов со 2 по 7 в палитру стандартной программы.





TObject — это начальная точка иерархии объектов Turbo Vision. Как базовый объект он не имеет предков, но имеет множество потомков. Кроме TPoint и TRect все стандартные объекты Turbo Vision порождены в конечном счете от TObject. Любой объект, использующий потоки Turbo Vision, должен происходить от TObject.

Методы

Init constructor Init;
Распределяет память в куче для объекта и заполняет его нулями. Вызывается из конструкторов всех порожденных объектов. Заметим, что TObject.Init будет заполнять все поля в потомках так, что вы должны вызвать TObject.Init до инициализации любых полей в конструкторах порожденных объектов.

Free procedure Free;
Освобождает объект и вызывает деструктор Done.

Done destructor Done; virtual;

Выполняет необходимую очистку и освобождение динамических объектов.

TParamText — Dialogs

TParamText порожден от TStaticText, который использует параметризованные строки текста для форматного вывода, используя процедуру FormatStr.

Поля

ParamCount ParamCount: Integer;

ParamCount указывает число параметров, содержащихся в ParamList.

См. также: TParamText.ParamList

ParamList ParamList: Pointer;

ParamList — это нетипизированный указатель на массив или запись указателей или значений типа LongInt, используемый как параметр форматирования для текстовой строки.

Методы

Init constructor Init(var Bounds: TRect; AText: String; AParamCount: Integer);

Инициализирует объект статического текста, вызывая TStaticText.Init с заданным Bounds и текстовой строкой AText, которая может содержать спецификаторы формата в форме % [-][nnn]X, который будет замещаться параметрами, переданными во время выполнения. Число параметров, переданных в ParamCount, присваивается полю ParamCount. Спецификаторы формата детально описаны в процедуре FormatStr.

См. также: TStaticText.Init, FormatStr процедуры

Load constructor Load(var S: TStream);

Распределяет объект TParamText в куче и загружает его значение из потока S, вызывая

TStaticText.Load, затем читая поле ParamCount из потока.

См. также: TStaticText.Load

DataSize function DataSize: Word; virtual;

Возвращает размер данных, требуемый параметрами объекта, т.е. ParamCount*SizeOf(LongInt).

GetText procedure GetText(var S: String); virtual;

Создает форматированную строку текста в S, выполняя объединение параметров, заданных в ParamList в текстовую строку Text с помощью вызова FormatStr(S, Text^, ParamList^).

См. также: FormatStr процедуры

SetData procedure SetData(var Rec); virtual;

Этот видимый элемент читает DataSize байт в ParamList из Rec.

См. также: TView.SetData

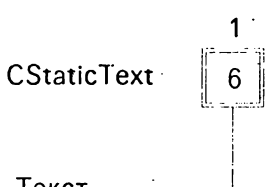
Store procedure Store(var S: TStream);

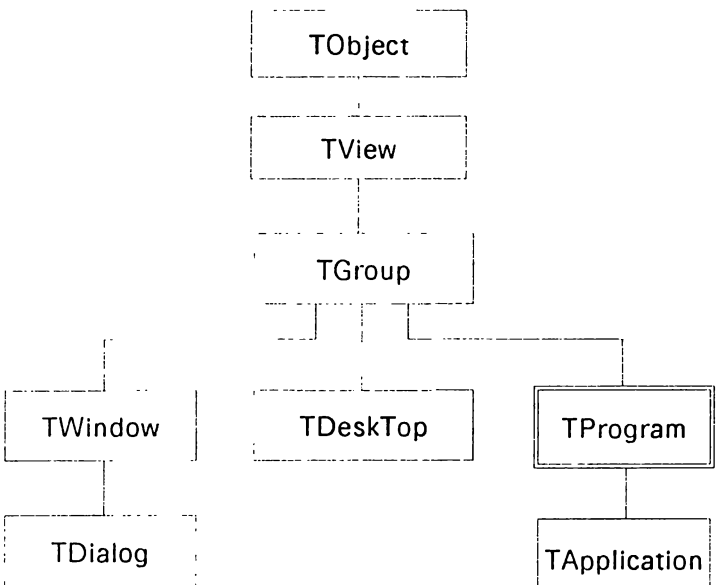
Сохраняет объект в потоке S, вызывая TStaticText.Store, затем записывая поле ParamCount в поток.

См. также: TStaticText.Store

Палитра

Объекты TParamText используют палитру по умолчанию CStaticText для отображения 6-го элемента в палитру стандартного диалога.





TPoint — это простой объект, представляющий точку на экране.

Поля

X X: Integer;
X — это колонка точки на экране.

Y Y: Integer;
Y — это строка точки на экране.

TProgram App

TProgram обеспечивает базовую заготовку для всех стандартных программ на Turbo Vision. Все такие программы должны порождаться от TProgram или его потомка TApplication. TApplication отличается от TProgram только конструктором и деструктором. Два типа объектов предоставляются для боль-

шей гибкости при создании нестандартных программ. В большинстве случаев ваша программа будет порождаться от TApplication. TProgram порождается от TGroup, поскольку она должна содержать объекты TDeskTop, TStatusLine и TMenuBar.

Методы

Init constructor Init; Перекрывается: Иногда
Устанавливает глобальную переменную Application в @Self; вызывает TProgram.InitScreen для инициализации переменных, определяющих режим экрана; вызывает TGroup.Init передавая прямоугольник Bounds, равный полному экрану; устанавливает поле State в sfVisible + sfSelected + sfFocused + sfModal + sfExposed; устанавливает поле Options в 0; устанавливает поле Buffer в адрес экранного буфера, заданного в ScreenBuffer; наконец, вызывает InitDeskTop, InitStatusLine и InitMenuBar и вставляет эти видимые элементы в группу TProgram.
См. также: TGroup.Init, TProgram.InitDeskTop, TProgram.InitStatusLine, TProgram.InitMenuBar

Done destructor Done; virtual; Перекрывается: Иногда
Освобождает объекты DeskTop, MenuBar и StatusLine и устанавливает глобальную переменную Application в nil.
См. также: TGroup.Done

GetEvent procedure GetEvent(var Event: TEvent); virtual;
Перекрывается: Редко
По умолчанию TView.GetEvent просто вызывает GetEvent своего владельца и, поскольку TProgram (или TApplication) в конечном итоге являются владельцем любого видимого элемента, любой вызов GetEvent будет приводить к TProgram. GetEvent (если только видимый элемент не перекроет GetEvent)
TProgram.GetEvent вначале проверяет, не сгенерировал ли TProgram.PutEvent событие. Если да, то GetEvent возвращает это событие. Если нет при-

шедшего события, GetEvent вызывает GetMouseEvent; если тот возвращает evNothing, вызывается GetKeyEvent, если оба возвращают evNothing: указывая, что нет ввода от пользователя, GetEvent вызывает TProgram.Idle, чтобы запустить «фоновые» задачи, выполняемые во время ожидания ввода от пользователя. До возврата GetEvent передает все события evKeyDown и evMouseDown в StatusLine для отображения в ассоциированные события от горячих клавиш evCommand.

См. также: TProgram.PutEvent, GetMouseEvent, GetKeyEvent

GetPalette function GetPalette: PPalette; virtual;

Перекрывается: Иногда

Возвращает указатель на палитру, заданную индексом палитры в глобальной переменной AppPalette. TProgram поддерживает 3 палитры: apColor, apBlackWhite и apMonochrome. Переменная AppPalette инициализируется в TProgram.InitScreen.

См. также: TProgram.InitScreen, AppPalette, apXXXX константы

HandleEvent procedure HandleEvent(var Event: TEvent); virtual;

Перекрывается: Всегда

Обрабатывает событие от клавиатуры от Alt-1 до Alt-9 генерируя событие evBroadcast с значением Command равным cmSelectWindowNum и значением InfoInt 1..9. TWindow.HandleEvent реагирует на такие сообщения, выбирая окно с данным номером. Обрабатывает событие evCommand со значением Command равным cmQuit вызывая EndModal(cmQuit) которое приводит к завершению программы.

TProgram.Handle почти всегда перекрывается для введения обработки команд, специфичных для вашей программы.

См. также: TGroup.HandleEvent

Idle procedure Idle; virtual; Перекрывается: Иногда

Idle вызывается из TProgram.GetEvent когда очередь событий пуста, позволяя программе выполнять фоновые задачи при ожидании ввода от пользователя. По умолчанию TProgram.Idle вызывает StatusLine^.Update, чтобы разрешить строке статуса обновлять себя в соответствии с текущим контекстом подсказки. Затем, если набор команд изменился после последнего вызова TProgram.Idle, генерируется evBroadcast со значением Command равным cmCommandSetChanged, чтобы разрешить видимому элементу, который зависит от этого набора команд, разрешить или запретить себя.

Если вы перекрываете Idle, всегда вызывайте наследуемый Idle. Так же убедитесь, что любые задачи, выполняемые в вашем Idle, не занимают слишком большого времени в программе, поскольку это будет блокировать ввод пользователя.

InitDeskTop procedure InitDeskTop; virtual;

Перекрывается: Редко

Создает объект TDeskTop для программы и сохраняет указатель на него в глобальной переменной DeskTop. InitDeskTop вызывается в TProgram.Init и никогда не должен вызываться прямо. InitDeskTop может быть перекрыт созданием потомка от TDeskTop.

См. также: TProgram.Init, TDeskTop, TWindow.Init

InitMenuBar procedure InitMenuBar; virtual;

Перекрывается: Всегда

Создает объект TMenuBar для программы и сохраняет указатель на него в глобальной переменной в MenuBar. InitMenuBar вызывается в TProgram.Init и никогда не должен вызываться прямо. InitMenuBar почти всегда перекрывается потомком от TMenuBar, определенным пользователем.

См. также: TProgram.Init, TMenuBar, TWindow.Init

InitScreen procedure InitScreen; virtual;

Перекрывается: Иногда

Вызывается из TProgram.Init и TProgram.SetScreenMode каждый раз, когда режим экрана инициализируется или изменяется. Это метод, который действительно выполняет обновление и настройку переменных, определяющих режим экрана для размера тени, маркеров и палитры программы.

См. также: TProgram.Init, TProgram.SetScreenMode

InitStatusLine procedure InitStatusLine; virtual;

Перекрывается: Всегда

Создает объект TStatusLine для программы и сохраняет указатель на него в глобальной переменной StatusLine. InitStatusLine вызывается в TProgram.Init и никогда не должен вызываться прямо. InitStatusLine почти всегда перекрывается потомком от TStatusLine, определенным пользователем.

См. также: TProgram.Init, TStatusLine

OutOfMemory procedure OutOfMemory; virtual;

Перекрывается: Часто

OutOfMemory вызывается из TProgram.ValidView для определения, что LowMemory — True. OutOfMemory должна сообщать пользователю, что недостаточно памяти для выполнения операции. Например с использованием программы MessageBox из модуля StdDlg:

```
procedure TMyApp.OutOfMemory;
begin
  MessageBox('Not enough memory to complete
    operation.', nil, mfError + mfOKButton);
end;
```

См. также: TProgram.ValidView, LowMemory

PutEvent procedure PutEvent(var Event: TEvent); virtual;

Перекрывается: Редко

По умолчанию TView.PutEvent просто вызывает PutEvent своего владельца и, поскольку объект TProgram (или TApplication) в конечном счете является владельцем любого видимого элемента, каждый вызов PutEvent будет приводить к TProgram.PutEvent (если только видимый элемент не перекрыл PutEvent).

Program.PutEvent сохраняет копию записи PutEvent в буфере и следующий вызов в TPreogram.GetEvent будет возвращать эту копию.

См. также: TProgram.GetEvent, TView.PutEvent

Run procedure Run; virtual; Перекрывается: Редко

Выполняет TProgram, вызывая метод Execute (который TProgram наследует от TGroup).

См. также: TGroup.Execute

SetScreenMode procedure SetScreenMode(Mode: Word);

Устанавливает режим экрана. Mode — одна из констант smCO80, smBW80 или smMono с дополнительным smFont8x8 для выбора 43- или 50-строчного режима на EGA или VGA. SetScreenMode скрывает мышку, вызывает SetVideoMode для изменения режима экрана, вызывает InitScreen для инициализации переменных режима экрана, назначает ScreenBuffer в TProgram.Buffer, вызывает ChangeBounds с новым прямоугольником экрана и, наконец, показывает мышку.

См. также: TProgram.InitScreen, SetVideoMode, smXXXX константа

ValidView function TProgram.ValidView(P: PView): PView;

Проверяет правильность вновь созданных видимых элементов, возвращая P, если видимый элемент правильный, и nil — если нет. Во-первых, если P — nil, возвращается значение nil. Во-вторых, если LowMemory — True, при вызове ValidView видимый элемент, заданный в P, освобождается, вызывается метод OutOfMemory и возвращается значе-

ние nil. В-третьих, если вызов P^.Valid(cmValid) возвращает False, видимый элемент освобождается и возвращается значение nil. Иначе видимый элемент считается правильным, и возвращается указатель на этот видимый элемент P.

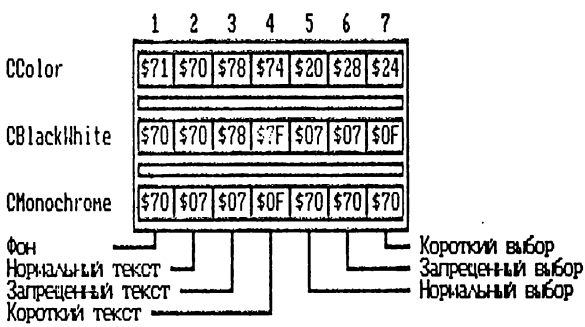
ValidView часто используется для проверки правильности нового видимого элемента до вставки его во владельца. Например, следующий оператор показывает типичную последовательность создания, проверки и вставки нового окна в панель экрана (TProgram.ValidView и TGroup.Insert знают как игнорировать возможные указатели nil, возникающие в результате ошибок).

```
DeskTop^.Insert(ValidView(New
                          (TMyWindow,Init(...))));
```

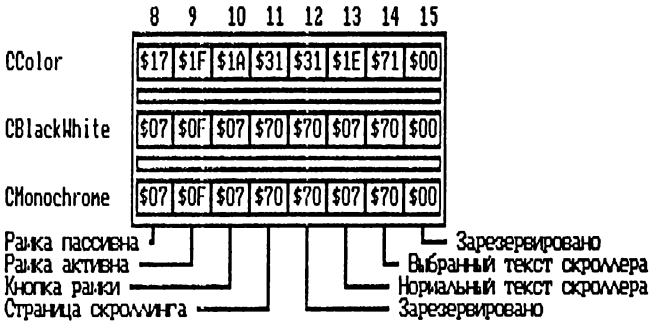
См. также: LowMemory, TProgram.OutOfMemory, Valid методы

Палитра

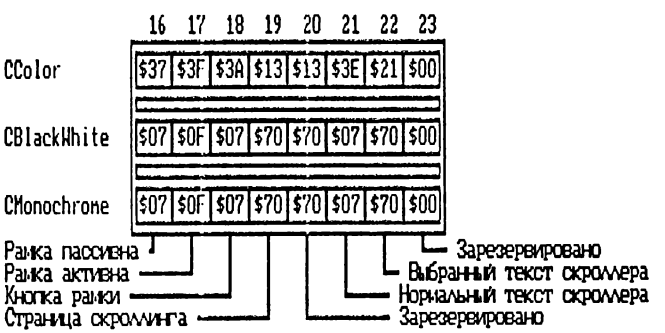
Палитра для объекта—программа управляет конечным отображением цвета всех видимых элементов программы. Все отображения других палитр приводят к выбору элемента в палитре программы, которая задаст атрибуты текста. Первый элемент используется в TBackground для цветов фона. Элементы со 2 по 7 используются меню и строкой статуса.



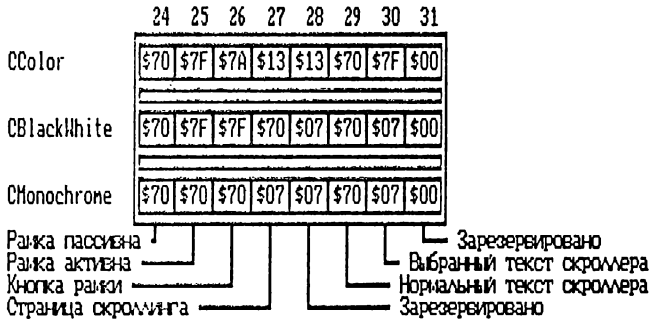
Элементы в 8 по 15 используются голубыми окнами.



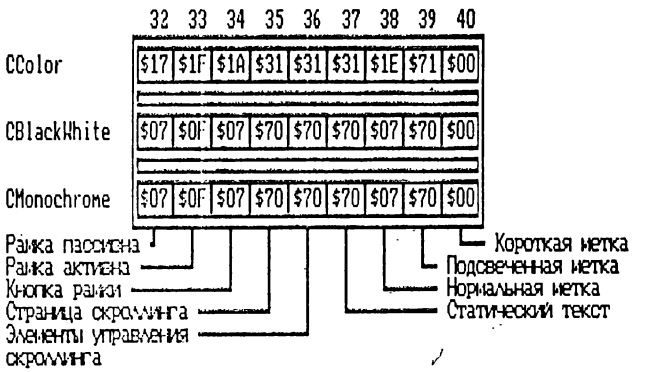
Элементы с 16 по 23 используются бирюзовыми окнами.

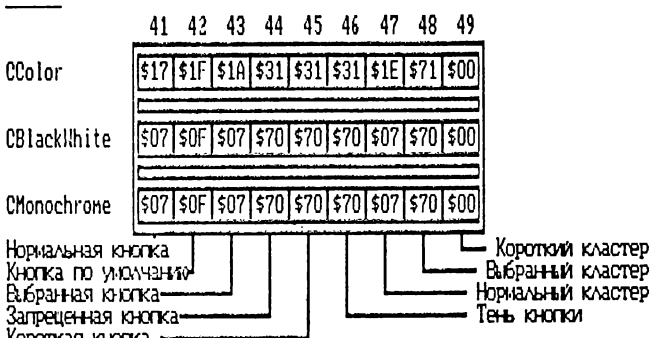


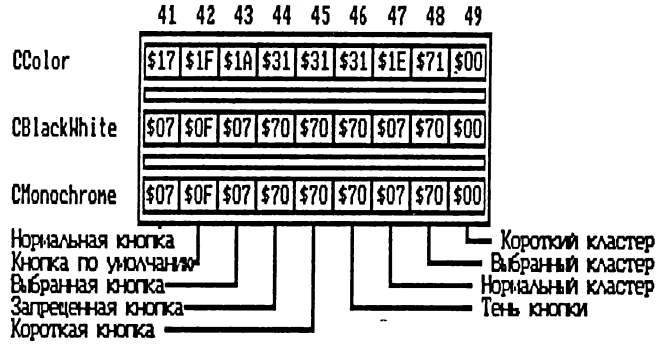
Элементы с 24 по 31 используются серыми окнами.

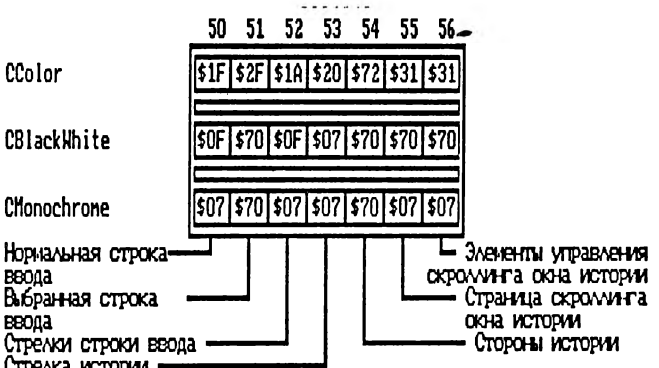


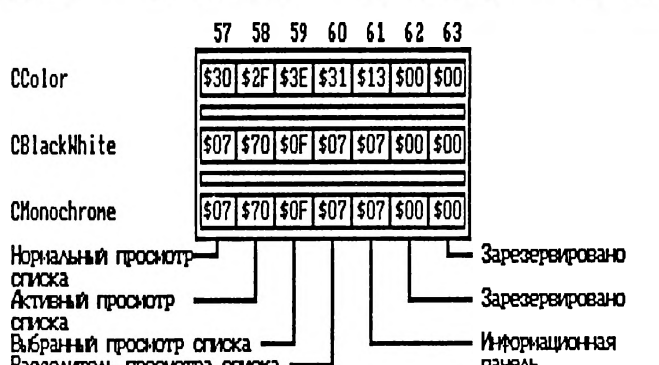
Элементы с 32 по 63 используются диалоговыми окнами. См. TDialog для отдельных элементов.

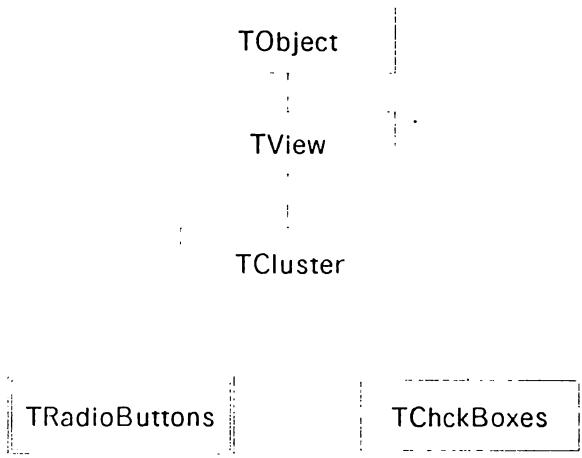





41 42 43 44 45 46 47 48 49
CColor
$17 $1F $1A $31 $31 $31 $1E $71 $00
CBlackWhite
$07 $0F $07 $70 $70 $70 $07 $70 $00
CMonochrome
$07 $0F $07 $70 $70 $70 $07 $70 $00
Нормальная кнопка
Кнопка по умолчанию
Выбранная кнопка
Запреценная кнопка
Короткая кнопка
Короткий кластер
Выбранный кластер
Нормальный кластер
Тень кнопки


50 51 52 53 54 55 56
CColor
$1F $2F $1A $20 $72 $31 $31
CBlackWhite
$0F $70 $0F $07 $70 $70 $70
CMonochrome
$07 $70 $07 $07 $70 $07 $07
Нормальная строка ввода
Выбранная строка ввода
Стрелки строки ввода
Стрелка истории
Элементы управления скроллинга окна истории
Страница скроллинга окна истории
Стороны истории


57 58 59 60 61 62 63
CColor
$30 $2F $3E $31 $13 $00 $00
CBlackWhite
$07 $70 $0F $07 $07 $00 $00
CMonochrome
$07 $70 $0F $07 $07 $00 $00
Нормальный просмотр списка
Активный просмотр списка
Выбранный просмотр списка
Разделитель просмотра списка
Зарезервировано
Зарезервировано
Информационная панель



Объекты TRadioButtons — это кластеры, содержащие до 65536 элементов управления, из которых в любой момент времени может быть выбрана только одна кнопка. Выбор невыбранной кнопки будет автоматически освобождать предварительно выбранную кнопку. Этот объект наследует от TCluster большую часть функций, включая Init, Load и Done. Зависимые кнопки часто ассоциированы с объектом TLabel. TRadioButtons интерпретирует наследуемое поле TCluster.Value как номер «нажатой» кнопки. С номером первой кнопки в кластере, равным 0.

Методы

Draw procedure Draw; virtual; Перекрывается: Редко
Рисует кнопки как ' ( ) '.

Mark function Mark(Item: Integer): Boolean; virtual;
Перекрывается: Никогда
Возвращает True, если Item = Value, т.е. если кнопка с номером Item представлена текущим значением поля Value.

См. также: TCluster.Value, TCluster.Mark

MovedTo procedure MovedTo(Item: Integer); virtual;

Перекрывается: Никогда

Присваивает Value значение Item.
См. также: TCluster.MovedTo,
TRadioButtons.Mark

Press procedure Press(Item: Integer); virtual;

Перекрывается: Никогда

Присваивает Value значение Item. Вызывается при нажатии кнопки с номером Item.

SetData procedure SetData(var Rec); virtual;

Перекрывается: Редко

Вызывает TCluster.SetData для установки поля Value, затем устанавливает поле Sel равным Value, поскольку выбранный элемент — это «нажатая» кнопка.
См. также: TCluster.SetData

Палитра

Объекты TRadioButtons используют CCluster — палитру по умолчанию для всех объектов кластера для отображения элементов с 16 по 18 в палитру стандартного диалога.

	1	2	3	4	
CCluster	16	17	18	18	
Нормальный текст					Короткая выбранная
Выбранный текст					Короткая нормальная

TRect — Objects

Поля

A A: TPoint

A — это точка, определяющая верхний левый угол прямоугольника на экране.

B B: Point

B — это точка, определяющая нижний правый угол прямоугольника на экране.

Методы

Assign procedure Assign(XA, YA, XB, YB: Integer);

Этот метод назначает значения параметров полям прямоугольника. XA становится A.X, XB становится X.B и т.д.

Copy procedure Copy(R: TRect);

Copy устанавливает все поля равными прямоугольнику R.

Move procedure Move(ADX, ADY: Integer);

Перемещает прямоугольник, добавляя ADX к A.X и B.X и добавляя ADY к A.Y и B.Y.

Grow procedure Grow(ADX, ADY: Integer);

Изменяет размер прямоугольника, вычитая ADX из A.X, добавляя ADX к B.X, вычитая ADY из A.Y и добавляя ADY к B. Y.

Intersect procedure Intersect(R: TRect);

Изменяет положение и размер прямоугольника до области, определенной пересечением текущего положения и R.

Union procedure Union(R: TRect);

Изменяет прямоугольник до его объединения с прямоугольником R; т.е. до наименьшего прямоугольника, содержащего этот объект и R.

Contains function Contains(P: TPoint): Boolean;

Возвращает True, если прямоугольник содержит точку P.

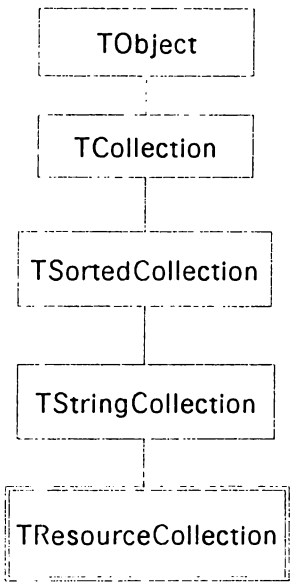
Equals function Equals(R: TRect): Boolean;

Возвращает True, если R равен данному прямоугольнику.

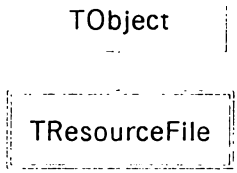
Empty function Empty: Boolean;

Возвращает True, если прямоугольник пустой, т.е. не содержит символьного пространства. Таким образом поля A и B равны.

TResourceCollection Objects



TResourseCollection порожден от TStringCollection и используется с TSourseFile для реализации коллекции ресурсов. Файл ресурсов — это поток, который индексируется ключевыми строками. Следовательно, каждый элемент ресурса имеет целое поле Pos и строковое поле Key. Перекрытие методов TResourseCollection главным образом связано с обработкой дополнительных строк в его элементах. TResourseCollection используется внутри объектов TResourseFile для поддержки индекса файла ресурсов.



TResourseFile реализует поток, который может индексироваться ключевыми строками. Когда объекты сохраняются в файле ресурса, используя TResourseFile.Put, задается ключевая строка, которая идентифицирует этот объект. Объект может быть позже получен указанием этой ключевой строки в вызове TResourseFile.Get.

Для обеспечения быстрого и эффективного доступа к объектам, хранящимся в файле ресурса, TResourseFile хранит ключевые строки в отсортированной коллекции строк (используя тип TResourseCollection) вместе с позицией и размером данных этого ресурса в файле ресурса. Как и в случае потоков, типы объектов, записываемые и читаемые из файла ресурсов, должны быть зарегистрированы с помощью RegisterType.

Поля

Stream Stream: PStream; Только чтение

Указатель на поток, связанный с этим файлом ресурса.

Modified Modified: Boolean; Чтение/Запись

Установлен в True, если файл ресурса был модифицирован.

Методы

Init constructor Init(AStream: Pstream);

Перекрывается: Никогда

Инициализирует файл ресурса, используя поток, заданный через AStream, и устанавливает поле Modified в False. Например:

```
ResFile.Init(New(TBufStream, Init('MYAPP.RES',
  stOpenRead, 1024)));
```

Во время инициализации Init смотрит в заголовке файла ресурсов текущую позицию в потоке. Формат заголовка файла ресурсов:

```
type
  TResFileHeader = record
  Signature: array[1..4] of Char;
  ResFileSize: Longint;
  IndexOffset: Longint;
end;
```

где Signature содержит 'FBPR', ResFileSize содержит размер всего файла ресурсов, за исключением полей Signature и ResFileSize (т.е. размер файла ресурса —8 байт) и IndexOffset содержит смещение коллекции индексов от начала заголовка.
Если Init не находит заголовка файла ресурса в текущей позиции AStream, он считает, что создается новый файл ресурса и создает пустой индекс.
Если Init видит метку .EXE файла в текущей позиции потока, он просматривает поток до конца файла .EXE и ищет заголовок файла ресурса здесь. Аналогично Init будет пропускать оверлейный файл, добавленный к .EXE файлу (также как OvrInit пропускает файл ресурса). Это означает, что вы можете добавить оверлейный файл и файл ресурса (в любом порядке) в конец .EXE файла вашей программы. (Именно это сделано с выполнимым файлом IDE — TURBO.EXE).
См. также: TResourceFile.Done

Done destructor Done; virtual; Перекрывается: Никогда
Выталкивает файл ресурса, используя TResourseFile.Flush, затем освобождает индекс и поток, указанный полем Stream.
См. также: TResourceFile.Init, TResourceFile.Flush

Count function Count: Integer;

Возвращает число ресурсов, запомненных в файле ресурсов.

См. также: TResourceFile.KeyOf

Delete procedure Delete(Key: String);

Удаляет ресурс, индексируемый ключом Key из файла ресурсов. Пространство, ранее занятое удаленным ресурсом, не используется. Вы можете удалить эту память, используя SwitchTo для создания упакованной копии файла в новом потоке.

См. также: TResourceFile.SwitchTo

Flush procedure Flush;

Если файл ресурса был модифицирован (проверяется поле Modified) Flush сохраняет обновленный индекс в конце потока и обновляет заголовок ресурса в начале потока. Затем Modified устанавливается в False.

См. также: TResourceFile.Done, TResourceFile.Modified

Get function Get(Key: String): PObject;

Ищет Key в индексе файла ресурсов. Возвращает nil, если ключ не найден. Иначе, устанавливает поток на позицию, заданную индексом и вызывает Stream^.Get для создания и загрузки объекта по этому индексу. Например

```
DeskTop^.Insert(ValidView(ResFile.Get
                    ('EditorWindow')));
```

См. также: TResourceFile.KeyAt, TResourceFile.Put

KeyAt function KeyAt(I: Integer): String;

Возвращает ключевую строку для ресурса с номером i в файле ресурса. Индекс первого ресурса 0 и индекс последнего ресурса TResourseFile.Count–1. Используя Count и KeyAt вы можете обработать все ресурсы в файле ресурса.

См. также: TResourceFile.Count

Put procedure Put(Item: PObject; Key: String);

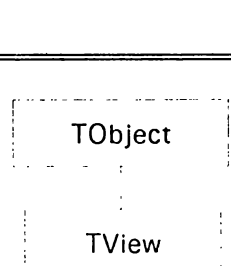
Добавляет объект, заданный через P, в файл ресурса с ключевой строкой, заданной в Key. Если индекс содержит Key, новый объект замещает старый. Объект добавляется в конец существующих объектов файла ресурса с использованием Stream^.Put

См. также: TResourceFile.Get

SwitchTo function SwitchTo(AStream: PStream;
Pack: Boolean): PStream;

Переключает файл ресурса из его потока в поток, переданный через AStream и возвращает указатель на новый поток. Если параметр Pack равен True, поток будет отбрасывать пустое и неиспользуемое пространство из файла ресурса до записи в новый поток. Это единственный способ сжать файл ресурса. Копирование с параметром Pack = False будет выполняться быстрее, но без сжатия.

TScrollBar Views



Поля

Value Value : Integer; Только чтение

Поле Value представляет текущую позицию индикатора полосы скроллинга. Этот маркер, выделенный цветом, перемещается по полосе скроллинга, указывая относительную позицию (горизонтальную или вертикальную в зависимости от полосы скроллинга) в тексте относительно всего текста,

доступного для скроллинга. Многие события могут прямо или косвенно изменять Value, такие, как отметки мышкой на элементах полосы скроллинга, изменение размера окна или изменение текста в скроллере. Аналогично, изменения в Value могут потребовать отображение в события. TScroolBar.Init устанавливает Value в 0.

См. также: TScrollBar.SetValue, TScrollBar.SetParams, TScrollBar.ScrollDraw, TScroller.HandleEvent, TScrollBar.Init

Min — Min: Integer; — Только чтение

Min представляет минимальное значение поля Value. По умолчанию TScrollBar устанавливает Min в 0.

См. также: TScrollBar.SetRange, TScrollBar.SetParams

Max — Max: Integer; — Только чтение

Max представляет максимальное значение поля Value. По умолчанию TScrollBar устанавливает Max в 0.

См. также: TScrollBar.SetRange, TScrollBar.SetParams

PgStep — PgStep: Integer; — Только чтение

PgStep — это количество добавляемое или вычитаемое из поля Value полосы скроллинга, когда событие от мышки возникает в любой части области страницы (sbPageLeft, sbPageRight, sbPageUp, sbPageDown) или обнаруженные эквивалентные клавиши (Ctrl-→, Ctrl-←, PgUp, PgDn). По умолчанию TScrollBar.Init устанавливает PgStep в 1. PgStep может изменяться при использовании TScrollBar.SetStep, TScrollBar.SetParams и TScroller.SetLimit.

См. также: TScrollBar.SetStep, TScrollBar.SetParams, TScroller.SetLimit, TScrollBar.ScrollStep

ArStep ArStep: Integer; Только чтение

ArStep — это количество, добавляемое или вычитаемое из поля Value полосы скроллинга когда отмечена область стрелок (sbLeftArrow, sbRightArrow, sbUpArrow, sbDownArrow) или обнаружены эквивалентные нажатия клавиш. По умолчанию TScrollBar.Init устанавливает ArStep в 1.

См. также: TScrolBar.SetStep, TScrollBar.SetParam, TScrollBarScrollStep

Методы

Init constructor Init(var Bounds: TRect);

Создает и инициализирует полосу скроллинга с границами Bounds, вызывая TView.Init. Value, Max и Min устанавливаются в 0. PgStep и ArStep устанавливаются в 1. Формы элементов полосы скроллинга по умолчанию устанавливаются в TScrollChars.

Если Bounds задает Size.X = 1, вы получите вертикальную полосу скроллинга, иначе — горизонтальную. Вертикальные полосы скроллинга имеют поле GrowMode, установленное в gfGrowLoX + gfGrowHiX + gfGrowHiY; вертикальные полосы скроллинга имеют поле GrowMode, установленное gfGrowLoY + gfGrouHiX + gfGrowHiY.

Load constructor Load(var S: TStrea m);

Создает и загружает полосу скроллинга из потока S, вызывая TView.Load, затем читая 5 целочисленных полей через S.Read.

См. также: TScrollBar.Store

Draw procedure Draw; virtual; Перекрывается: Никогда

Рисует полосу скроллинга в зависимости от текущих Bounds, Value и палитры.

См. также: TScrollBar.ScrollDraw, TScrollBar.Value

GetPalette function GetPalette: PPalette; virtual;

Перекрывается: Иногда

Возвращает указатель на CScrollBar, палитру по умолчанию для полосы скроллинга.

HandleEvent procedure HandleEvent(var Event: TEvent); virtual;

Перекрывается: Никогда

Обрабатывает события полосы скроллинга, вызывая TView.HandleEvent, затем анализируя Event.What. События от мышки — это общие сообщения владельцу полосы скроллинга (см. функцию Message), которые должны обрабатываться неявными изменениями полосы скроллинга, например скроллинг текста. TScrollBar.HandleEvent также определяет, какая часть полосы скроллинга получила отметку от мышки (или эквивалентную клавишу). Поле Value настраивается в соответствии с текущими значениями ArStep и PgStep и индикатор полосы скроллинга перерисовывается.

См. также: TView.HandleEvent

ScrollDraw procedure ScrollDraw; virtual;

Перекрывается: Редко

ScrollDraw вызывается при изменении поля Value. Этот псевдоабстрактный метод вызывается передачей сообщения cmScrollBarChanged владельцу полосы скроллинга:

```
Message(Owner, evBoadcast,
        cmScrollBarChanged, @Self);
```

См. также: TScrollBar.Value, Message функция

ScrollStep function ScrollStep(Part: Integer): Integer; virtual;

Перекрывается: Никогда

По умолчанию ScrollStep возвращает положительное или отрицательное значение шага в зависимости от части полосы скроллинга, заданной в Part и текущих значений ArStep и PgStep. Аргумент Part должен быть одной из констант sbXXXX описанных в главе 4.

См. также: TScrollBar.SetStep, TScrollBar.SetParams

SetParams procedure SetParams(AValue,AMin, AMax, APgStep, AArStep: Integer);

SetParams устанавливает поля Value, Min, Max, PgStep и ArStep в заданные значения. Если аргументы конфликтуют, выполняются согласования. Например, Min не может быть больше Max, поэтому, если AMax < AMin, Max устанавливается в Min. Value должно лежать в диапазоне [Min, Max], поэтому, если Value < AMin, Value устанавливается в Min; если AValue > AMax, Value устанавливается в Max. DrawView перерисовывает полосу скроллинга. Если Value изменяется, будет вызвана ScroolDraw.

См. также: TView.DrawView, TScrollBar.ScrollDraw, TScrollBar.SetRange, TScrollBar.SetValue

SetRange procedure SetRange(AMin, AMax: Integer);

SetRange задает допустимый диапазон для поля Value, устанавливая Min и Max в AMin и AMax. SetRange вызывает SetParams, поэтому DrawView и ScroolBar будут вызываться, если изменения требуют перерисовки полосы скроллинга.

См. также: TScrollBar.SetParams

SetStep procedure SetStep(APgStep, AArStep: Integer);

SetStep устанавливает поля PgStep и ASrStep в APgStep и AArStep. Этот метод вызывает SetParams с остальными аргументами, равными их текущим значениям.

См. также: TScrollBar.SetParams, TScrollBar.ScrollStep

SetValue procedure SetValue(AValue: Integer);

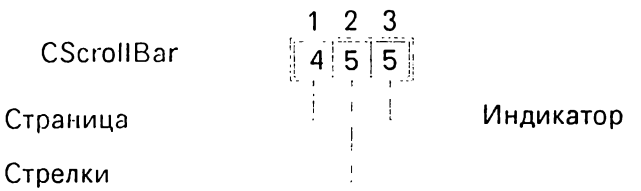
SetValue устанавливает поле Value в AValue, вызывая SetParams с остальными аргументами, установленными в их текущие значения. DrawView и ScrollDraw вызываются, если этот вызов изменяет значение Value.

См. также: TScroll Bar.SetParams, TView.DrawView, TScrollBar.ScrollDraw, TScroller.ScrollTo

Store procedure Store(var S:TStream);
Сохраняет объект TScrollBar в потоке S, вызывая TView.Store, затем записывая 5 целочисленных полей в поток, используя S.Write.
См. также: TScrollBar.Load

Палитра

Объекты полосы скроллинга используют палитру по умолчанию CScrollBar для отображения в 4 и 5-й элементы палитры стандартной программы.



TScroller — Views

TObject

TView

TScroller

TTexDevise

TTerminal

Поля

HScrollBar HScrollBar:PScrollBar; Только чтение

Указывает на горизонтальную полосу скроллинга, связанную с этим скроллером. Если такой полосы скроллинга нет, HScroolBar равен nil.

VScrollBar VScrollBar: PScrollBar; Только чтение

Указывает на вертикальную полосу скроллинга, связанную с этим скроллером. Если такой полосы скроллинга нет, VScroolBar равен nil.

Delta Delta: TPoint; Только чтение

Содержит X (горизонтальная) и Y (вертикальная) компоненты позиции скроллера относительно виртуального видимого элемента. Автоматический скроллинг достигается изменением одной или обоих компонент в ответ, например, на события полосы скроллинга, изменяющих значения поля Value. Ручной скроллинг изменяет Delta, отображает изменения в поле Value полосы скроллинга и приводит к обновлению индикаторов полосы скроллинга.

См. также: TScroller.ScrollDraw, TScroller.ScrollTo

Limit Limit: TPoint; Только чтение

Limit.X Limit.Y — это максимально допустимые значения для Delta.X и Delta.Y.

См. также: TScroller.Delta

Методы

Init constructor Init(var Bounds: TRect; AHScrollBar, AVScrollBar: PScrollBar);

Создает и инициализирует объект TScroller с заданным размером и полосами скроллинга. Вызывает TView.Init для установки размера видимого элемента. Options устанавливается в ofSelectabble, а EventMask устанавливается в evBroadcast. AHScrollBar должен быть nil, если вы не хотите горизонтальную полосу скроллинга; аналогично AVScrollBar должен быть nil, если вы не хотите вертикальной полосы скроллинга.

См. также: TView.Init, TView.Options, TView.EventMask

Load constructor Load(var S: TStream);

Загружает видимый элемент скроллера из потока S, вызывая TView.Load, затем восстанавливает указатели на полосы скроллинга, используя GetPeerViewPtr и читает поля Delta и Limit, используя S.Read.

См. также: TScroller.Store

ChangeBounds procedure ChangeBounds(var Bounds: TRect); virtual;

Перекрывается: Никогда

Изменяет размер скроллера, вызывая SetBounds. Если необходимо, скроллер и полосы скроллинга перерисовываются вызовом DrawView и SetLimit.

См. также: TView.SetBounds, TView.DrawView, TScroller.SetLimit

GetPalette function GetPalette: PPalette; virtual;

Перекрывается: Иногда

Возвращает указатель на палитру скроллера по умолчанию CScroller.

HandleEvent procedure HandleEvent(var Event: TEvent); virtual;

Перекрывается: Редко

Обрабатывает большинство событий, вызывая TView.HandleEvent. Общие события с командой cmScrollBarChanged, если они пришли от HScrollBar или VScrollBar, приводят к вызову TScroller.ScrollDraw.

См. также: TView.HandleEvent, TScroller.ScrollDraw

ScrollDraw procedure ScrollDraw; virtual;

Перекрывается: Никогда

Проверяет, соответствует ли Delta соответствующим позициям полос скроллинга. Если нет — Delta устанавливается в корректное значение и вызывается DrawView для перерисовки скроллера.

См. также: TView.DrawView, TScroller.Delta, TScroller.HscrollBar, TScroller.VScrollBar

ScrollTo procedure ScrollTo(X, Y: Integer);

Устанавливает полосы скроллинга в (X, Y), вызывая HScrollBar^.SetValue(X) и VScrollBar^.SetValue(Y) и перерисовывает видимый элемент, вызывая DrawView.

См. также: TView.DrawView, TScroller.SetValue

SetLimit procedure SetLimit(X, Y: Integer);

Устанавливает Limit.X в X и Limit.Y в Y, затем вызывает HScrollBar^.SetParams и VScrollVar^.SetParams (если эти полосы скроллинга существуют), чтобы настроить их поля Max. Эти вызовы могут привести к перерисовке полосы скроллинга. Наконец вызывается DrawView для перерисовки скроллера, если это необходимо.

См. также: TScroller.Limit, TScroller.HScroller, TScroller.VScrollBar, TScrollBar.SetParams

SetState procedure SetState(AState: Word;
Enable: Boolean); virtual;

Перекрывается: Редко

Этот метод вызывается при изменении состояния скроллера. Вызов TView.SetState устанавливает или очищает флаги состояния в State. Если новое состояние — sfSelected и sfActive, SetState отображает полосы скроллинга, иначе они скрываются.

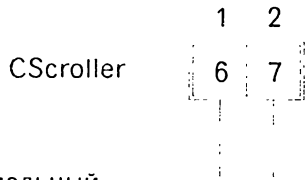
Store procedure Store(var S: TStream);

Записывает скроллер в поток S, вызывая TView.Store, затем сохраняет ссылки на полосы скроллинга, используя PutPeerViewPtr, наконец записывает значения Delta и Limit, используя S.Write.

См. также: TScroller.Load, TStream.Write

Палитра

Объекты скроллера используют палитру по умолчанию CScroller для отображения в 6 и 7 элементы палитры стандартной программы.



TObject

TCollection

TSortedCollection

TStringCollection

TResourceCollection

TSortedCollection порожден от TCollection и реализует коллекцию, отсортированную по ключу без дублирования. Сортировка производится методом TStringCollection.Compare, который вы перекрываете, чтобы задать свое определение упорядочивания элементов. Когда новые элементы добавляются, они автоматически вставляются в порядке, заданном методом Compare. Элементы будут располагаться, используя двоичный метод поиска

TStringCollection.Search. Виртуальный метод KeyOf, возвращающий указатель для Compare, также может быть перекрыт, если Compare требует дополнительной информации.

Методы

Compare function Compare(Key1, Key2: Pointer): Integer; virtual;

Перекрывается: Всегда

Compare — это абстрактный метод, который должен быть перекрыт во всех порожденных типах. Compare должен сравнивать два ключевых значения и возвращать результат:

```
-1 if Key1 < Key2
 0 if Key1 = Key2
 1 if Key1 > Key2
```

Key1 и Key2 — это значения указателей, извлеченных из соответствующей коллекции элементов методом TSortedCollection.KeyOf. Метод TSortedCollection.Search реализует двоичный поиск элементов коллекции, используя Compare для сравнения элементов.

См. также: TSortedCollection.KeyOf, TSortedCollection.Compare

IndexOf function IndexOf(Item: Pointer): Integer; virtual;

Перекрывается: Никогда

Использует TSortedCollection.Search для нахождения индекса элемента Item. Если элемент не в коллекции, IndexOf возвращает -1. Реализация TSortedCollection.IndexOf:

```
if Search(KeyOf(Item), I)
then IndexOf := I else IndexOf := -1;
```

См. также: TSortedCollection.Search

Insert procedure Insert(Item: Pointer); virtual;

Перекрывается: Никогда

Если элемент не найден в коллекции, он вставляется в позицию, определенную индексом, вызывает TSortedCollection.Search для определения, сущест-

вует ли элемент. Если нет, куда вставить его. Реализация TSortedCollection.Insert:

```
if not Search(KeyOf(Item), I) then AtInsert(I, Item)
```

См. также: TSortedCollection.Search

KeyOf `function KeyOf(Item: Pointer): Pointer; virtual;`

Перекрывается: Иногда

Для данного элемента коллекции KeyOf возвращает соответствующий ключ элемента. По умолчанию TSortedCollection.KeyOf просто возвращает Item. KeyOf перекрывается в случае, когда ключ элемента не совпадает с элементом.

См. также: TSortedCollection.IndexOf

Search `function Search(Key: Pointer; var Index: Integer): Boolean; virtual;`

Перекрывается: Редко

Возвращает True, если элемент, заданный ключом Key, не найден в отсортированной коллекции. Если элемент найден, Index устанавливается в найденный индекс; иначе Index устанавливается в индекс, куда будет помещаться элемент при вставке.

См. также: TSortedCollection.Compare, TSortedCollection.Insert

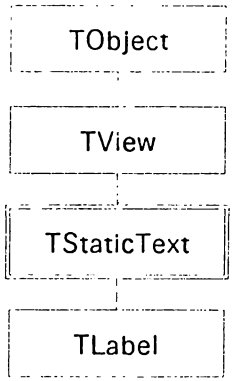
TStaticText — Dialogs

Объекты TStaticText представляют простейшие видимые элементы: они содержат фиксированный текст и игнорируют все события, переданные им. Они используются как сообщения или пассивные метки. Наследники TStaticText выполняют более активную роль.

Поля

Text `Text: PString;` Только чтение

Указатель на строку текста, отображаемую в видимом элементе.



Методы

Init constructor Init(var Bounds: TRect; AText: String);
Создает объект StaticText заданного размера, вызывая TView.Init, затем устанавливая текст в NewStr(AText).
См. также: TView.Init

Load constructor Load(varS: TStream);
Создает и инициализирует объект TStaticText из данного потока. Вызывает TView.Load и устанавливает текст с помощью S.ReadStr. Используется совместно с TStaticText.Store для сохранения и чтения статического текстового видимого элемента из потока.
См. также: TViewLoad, TStaticText.Store, TStream.ReadStr

Done destructor Done; virtual; Перекрывается: Редко
Освобождает строку Text, затем вызывает TView.Done для разрушения объекта.

Draw procedure Draw; virtual; Перекрывается: Редко
Рисует строку текста внутри видимого элемента, слово при необходимости переносится. Ctrl-M в тексте указывает на начало новой строки. Если строка начинается с Ctrl-C, она центрируется в видимом элементе.

GetPalette function GetPalette: PPalette; virtual;

Перекрывается: Иногда

Возвращает указатель на палитру по умолчанию CStaticText.

GetText procedure GetText(varS: String); virtual;

Перекрывается: Иногда

Возвращает в S строку, на которую указывает Text.

Store procedure TStaticText.Store(var S: TStream);

Сохраняет объект TStaticText в потоке, вызывая TView.Store и S.WriteStr. Используется совместно с TStaticText.Store для сохранения и чтения статического текстового видимого элемента из потока.

См. также: TStaticText.Load, TView.Store, TStream.WriteStr

Палитра

Статический текст использует палитру по умолчанию CStaticText для отображения в 6-й элемент палитры стандартного диалога.

	1
CStaticText	6

Цвет текста

TStatusLine — Menus

TObject

TView

TStatusLine

Объект TStatusLine — это видимый элемент, обычно отображаемый внизу экрана. Типичная строка статуса отображает список доступных горячих клавиш, свободную память, время дня, текущий режим редактирования и подсказки пользователя. Отображаемые элементы устанавливаются в связанный список, используя InitStatusLine в TApplication и отображаемый элемент зависит от контекста подсказки текущего видимого элемента. Как и полоса меню и панель экрана, строка статуса обычно принадлежит группе TApplication.

Элементы строки статуса — это записи типа TStatusItem, которые содержат поля для текстовой строки, отображаемой в строке статуса, кода ключа, связываемого с горячей клавишей (обычно функциональная клавиша или комбинация Alt -клавиша) и команды, генерируемой, если отображаемый текст отмечен мышкой или нажата горячая клавиша.

Строка статуса отображает контекстно-ориентированную подсказку. Каждый объект строки статуса содержит связанный список строк статуса Defs (типа TStatusDef), которые определяют диапазон контекстных подсказок и список элементов статуса, отображаемый, когда текущий контекст подсказки находится в этом диапазоне. Кроме того, может отображаться предопределенная строка в соответствии с текущим контекстом подсказки.

Поля

Items Items: PStatusItem; Только чтение

Указатель на текущий связанный список записей типа TStatusItem.

См. также: TStatusItem

Defs Defs: PStatusDef; Только чтение

Указатель на текущий связанный список записей типа TStatusDef. Список для использования определяется текущим контекстом подсказки.

См. также: TStatusDef, TStatusLine.Update, TStatusLine.Hint

Методы

Init
```
constructor Init(var Bounds: TRect;
                 ADefs: PStatusDef);
```
Создает объект TStatusLine с размером Bounds, вызывая TView.Init. Бит ofPreProcess в Options устанавливается, EventMask устанавливается, включая evBroadcast и GrowMode устанавливается в gfGrowLoY + gfGrowHiX + gfGrowHiY. Поле Defs устанавливается в ADefs. Если ADefs — nil, Items устанавливается в nil, иначе Items устанавливается в ADefs^.Items. См. также: TView.Init

Load
```
constructor Load(var S: TStream);
```
Создает объект TStatusLine и загружает его из потока S, вызывая TView.Load, затем читая Defs и Items из потока.
См. также: TView.Load, TStatusLine.Store

Done
```
destructor Done; virtual;
```
Перекрывается: Никогда
Освобождает все Items и Defs в объекте TStatusLine, затем вызывает TView.Done.
См. также: TView.Done

Draw
```
procedure Draw; virtual;
```
Перекрывается: Редко
Рисует строку статуса, выводя строку Text для каждого элемента статуса, затем все подсказки, определенные для данного текущего контекста подсказки за полосой разделителя.
См. также: TStatusLine.Hint

GetPalette
```
function GetPalette: PPalette; virtual;
```
Перекрывается: Иногда
Возвращает указатель на палитру по умолчанию CStatusLine.

HandleEvent
```
procedure HandleEvent(var Event: TEvent);
virtual;
```
Перекрывается: Редко
Обрабатывает события, передаваемые строке статуса, вызывая TView.HandleEvent, затем проверя-

ст на три вида специальных событий. Отметки мышкой, которые попадают внутрь прямоугольника, занимаемого элементом статуса, генерируют командное событие с Event.What, установленным в Command, для этого элемента статуса. События от клавиатуры сравниваются с полем KeyCode каждого элемента; соответствие вызывает командное событие с Command этого элемента. Общие события с командой cmCommand, SetChanged заставляют строку статуса перерисовывать себя, чтобы отразить любые горячие клавиши, которые могут быть разрешены или запрещены.

См. также: TView.HandleEvent

Hint `function Hint(AHelpCtx: Word): String; virtual;`

Перекрывается: Часто

Этот псевдоабстрактный метод возвращает пустую строку. Он должен быть перекрыт для обеспечения строки контекстно-ориентированной подсказки для аргумента AHelpCtx. Непустая строка будет рисоваться в строке статуса после полосы разделителя. См. также: TStatusLine.Draw

Store `procedure Store(var S: TStream);`

Сохраняет объект TStatusLine в потоке S, вызывая TView.Store, затем записывая все определения статуса и их ассоциированные списки элементов в поток. Сохраненный объект может быть восстановлен используя TStatusLine.Load.

См. также: TView.Store, TStatusLine.Load

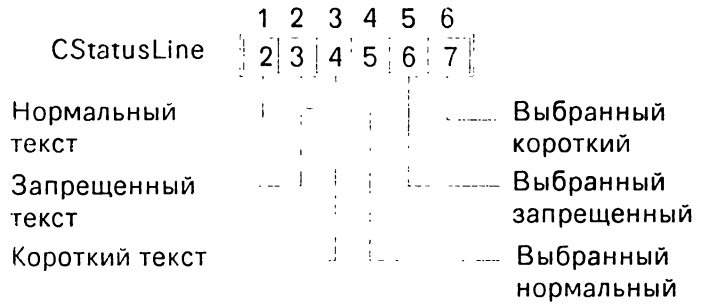
Update `procedure Update;`

Выбирает корректный Items из списка Defs. В зависимости от текущего контекста подсказки, затем вызывает DrawView для перерисовки строки статуса, если элементы были изменены.

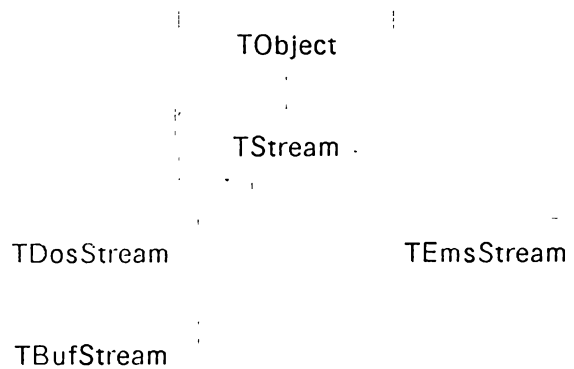
См. также: TStatusLine.Defs

Палитра

Строки статуса используют палитру по умолчанию CStatusLine для отображения в элементы со 2 по 7-й в палитре стандартной программы.



TStream — Objects



TStream — это общий абстрактный объект, обеспечивающий полиморфический ввод-вывод в и/из устройства памяти. Вы можете создать порожденный объект потока, перекрывая виртуальные методы GetPos, GetSize, Read, Seek, Truncate и Write. Turbo Vision делает это в порожденных потоках TDosStream и TEmsStream. Для порожденного буферизованного потока вы должны также перекрыть TStream.Flush.

Поля

Status `Status: Integer;` — Чтение/Запись

Указывает текущий статус потока:

Таблица 3.1.

Коды ошибок потока

Коды ошибок	TStream
stOk	Нет ошибок
stError	Ошибка доступа
stInitError	Нельзя инициализировать поток
stReadError	Чтение за концом файла
stWriteError	Нельзя расширить поток
stGetError	Get для незарегистрированного типа
stPutError	Put для незарегистрированного типа

Если Status <> stOK, все операции над потоком будут запрещены до тех пор, пока не будет вызван Reset.

ErrorInfo ErrorInfo: Integer; Чтение/Запись

Содержит дополнительную информацию когда Status не stOK. Для значений Status: stError, stInitError, stReadError, stWriteError, ErrorInfo содержит код ошибки DOS или EMS, если такой существует. Когда Status stGetError, ErrorInfo содержит IDE типа объекта (поле ObjType в TStreamRec) не зарегистрированного типа объекта. Когда Status — stPutError, ErrorInfo содержит смещение VMT в сегменте данных (поле VmtLink в TStreamRec) не зарегистрированного типа объекта.

Методы

CopyFrom procedure CopyFrom(var S: TStream;
Count: Longint);

Копирует Count байт из потока S в вызывающий поток. Например:

```
NewStream := New(TEmsStream,
Init(OldStream^.GetSize));
OldStream^.Seek(0);
NewStream^.CopyFrom(OldStream,
OldStream^.GetSize);
```

См. также: TStream.GetSize, TObject.Init

Error procedure Error(Code, Info: Integer); virtual;
Перекрывается: Иногда
Вызывается, если возникла ошибка потока. По умолчанию TStream.Error сохраняет Code и Info в полях Status и ErrorInfo. Затем, если глобальная переменная StreamError не nil, вызывает процедуру, заданную в StreamError. После возникновения ошибки, все операции над потоком запрещены до тех пор, пока не будет вызван Reset.
См. также: TStream.Reset, StreamError переменная

Flush procedure Flush; virtual; Перекрывается: Иногда
Абстрактный метод, который должен быть перекрыт, если ваш порожденный тип реализует буфер. Этот метод может выталкивать любые буфера, очищая буфер чтения и записывая буфер вывода. По умолчанию TStream.Flush ничего не делает.
См. также: TDosStream.Flush

Get function Get: PObject;
Читает объект из потока. Объект должен быть предварительно записан в поток через TStream.Put. Get вначале читает ID типа объекта (слово) из потока. Затем он находит соответствующий тип объекта, сравнивая ID с полем ObjType всех зарегистрированных типов объектов (см. тип TStreamRec). Наконец, вызывает конструктор Load этого типа объекта для создания и загрузки объекта. Если ID типа объекта, считанного из потока, равен 0, Get возвращает указатель nil; если ID типа объекта не зарегистрирован (используя RegisterType) Get вызывает TStream.Error и возвращает указатель nil; иначе Get возвращает указатель на вновь созданный объект.
См. также: TStream.Put, RegisterType, TStreamRec, Load методы

GetPos function GetPos: Longint; virtual;
Перекрывается: Всегда

Возвращает текущую позицию в потоке. Этот абстрактный метод должен всегда перекрываться.
См. также: TStream.Seek

GetSize function GetSize: Longint; virtual;

Перекрывается: Всегда

Возвращает размер потока. Это абстрактный метод и должен перекрываться.

Put procedure Put(P: PObject);

Записывает объект в поток. Объект позже можно считать из потока, используя TStream.Get. Put вначале находит регистрационную запись типа этого объекта, сравнивая смещение VMT объекта с полем VmtLink всех зарегистрированных типов объектов (см. тип TStreamRec). Затем записывает ID типа объекта (поле ObjType регистрационной записи) в поток, и наконец вызывает метод Store этого типа объекта для записи объекта. Если аргумент P, переданный в Put — nil, Put записывает в поток слово, содержащее 0. Если тип объекта в P не зарегистрирован (используя RegisterType), Put вызывает TStream.Error и ничего не пишет в поток.
См. также: TStream.Get, RegisterType, TStreamRec, Store методы

Read procedure Read(var Buf; Count: Word); virtual;

Перекрывается: Всегда

Это абстрактный метод и должен перекрываться во всех порожденных типах. Read должен читать Count байт из потока в Buf и перемещать текущую позицию потока на Count байт. Если произошла ошибка, Read должен вызывать Error и заполнять Buf Count байтами, равными 0.
См. также: TStream,Write, TStream.Error

ReadStr function ReadStr: PString;

Читает строку из текущей позиции потока, возвращая указатель PString. TStream.ReadStr вызывает GetMem для распределения (Length+1) байт для строки. См. также: TStream.WriteStr

Reset procedure Reset;

Сбрасывает ошибочное условие потока, устанавливая Status и ErrorInfo в 0. Этот метод позволяет вам продолжать обработку потока после ошибочной ситуации, которую вы скорректировали.

См. также: TStream.Status, TStream.ErrorInfo, sfXXXX коды ошибок

Seek procedure Seek(Pos: Longint); virtual;

Перекрывается: Всегда

Это абстрактный метод и должен перекрываться во всех потомках. TStream.Seek устанавливает текущую позицию в Pos байт, начиная от начала потока. Начало потока — позиция 0.

См. также: TStream.GetPos

Truncate procedure Truncate; virtual;

Перекрывается: Всегда

Это абстрактный метод и должен перекрываться во всех потомках. TStream.Truncate удаляет все данные в потоке от текущей позиции до конца.

См. также: TStream.GetPos, TStream.Seek

Write procedure Write(var Buf; Count: Word); virtual;

Перекрывается: Всегда

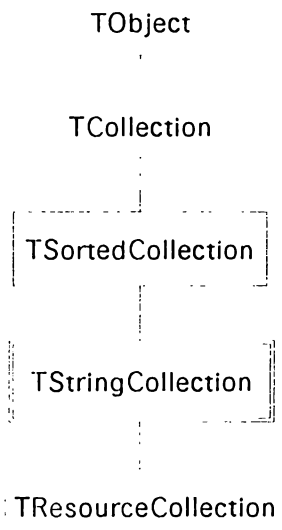
Это абстрактный метод и должен перекрываться во всех потомках. Write записывает Count байт из Buf в поток и перемещает текущую позицию потока на Count байт. Если возникла ошибка, Write должен вызывать Error.

См. также: TStream.Read, TStream.Error

WriteStr procedure WriteStr(P: PString);

Записывает строку P^ в поток, начиная с текущей позиции.

См. также: TStream.ReadStr



TStringCollection порожден от TSortedCollection и реализует сортированный список ASCII строк. Метод TStringCollection.Compare перекрывается для задания обычного лексикографического упорядочения строк ASCII. Вы можете перекрыть Compare для задания другого упорядочения, такого как для неанглийских наборов символов.

Методы

Compare function Compare(Key1, Key2: Pointer): Integer; virtual;

Перекрывается: Иногда

Сравнивает строки Key1^ и Key2^: возвращает −1, если Key1 < Key2; 0, если Key1 = Key2 и +1, если Key1 > Key2.

См. также: nTStringCollection.Search

FreeItem procedure FreeItem(Item: Pointer); virtual;

Перекрывается: Редко

Удаляет строку Item^ из отсортированной коллекции и освобождает строку.

GetItem function GetItem(var S: TStream): Pointer; virtual;

Перекрывается: Редко

По умолчанию читает строку из TStream, вызывая SReadStr.

См. также: TStream.ReadStr

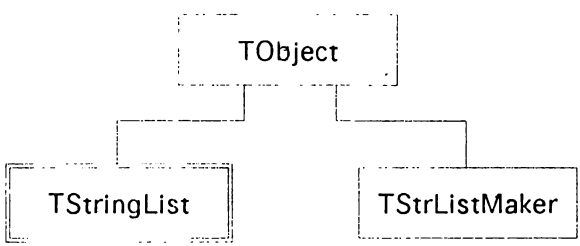
PutItem procedure PutItem(var S: TStream; Item: Pointer); virtual;

Перекрывается: Редко

По умолчанию записывает строку Item^ в TStream, вызывая S. WriteStr.

См. также: TStream.WriteStr

TStringList — Objects



TStringList предоставляет механизм для доступа к строкам, хранящимся в потоке. Каждая строка, хранящаяся в списке строк идентифицируется уникальным номером (ключом) между 0 и 65,535. Списки строк занимают меньше памяти, чем обычные строки, поскольку строки хранятся в потоке, а не в памяти. Кроме того, списки строк легко решают проблему настройки программ на языки, поскольку строки не «встроены» в программу.

TStringList имеет методы только для доступа к строкам; для создания списка строк вы должны использовать TStrListMaker.

Заметим, что TStringList и TStrListMaker имеют один ID типа объекта (поле ObjType в TStreamRec) и следовательно, не могут регистрироваться и использоваться одновременно в одной программе.

Методы

Load constructor Load(var S:TStream);

Загружает индекс списка строк из потока S и хранит ссылку на S так, что TStringLis.Get может обращаться к потоку при чтении строк.

Считая, что TStringList был зарегистрирован, используя RegisterType(RStringList), здесь показано, как считать список строк (созданный с использованием TStrListMaker и TResourseFile.Put) из файла ресурса:

```
ResFile.Init(New(TBufStream, Init('MYAPP.RES',
  stOpenRead, 1024)));
Strings := PStringList(ResFile.Get('Strings'));
```

См. также: TStrListMaker.Init, TStringList.Get

Done destructor Done; virtual; Перекрывается: Никогда

Освобождает память, распределенную под список строк.

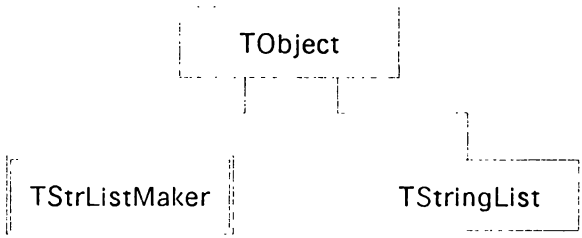
См. также: TStrListMaker.Init, TStringList.Done

Get function Get(Key: Word): String;

Возвращает строку, заданную через Key, или пустую строку, если нет строки с данным Key. Например:

```
P := @FileName;
FormatStr(S, Strings^.Get(sLoadingFile), P);
```

См. также: TStringListMaker.Put



TStrListMaker — это простой тип объекта, используемый для создания списка строк, который используют с TStringList. Следующий фрагмент кода показывает, как создавать и сохранять список строк в файле ресурса.

```
const
  sInformation = 100;
  sWarning = 101;
  sError = 102;
  sLoadingFile = 200;
  sSavingFile = 201;

var
  ResFile: TResourceFile;
  S: TStrListMaker;

begin
  RegisterType(RStrListMaker);
  ResFile.Init(New(TBufStream, Init('MYAPP.RES',
    stCreate, 1024)));
  S.Init(16384, 256);
  S.Put(sInformation, 'Information');
  S.Put(sWarning, 'Warning');
  S.Put(sError, 'Error');
  S.Put(sLoadingFile, 'Loading file #s.');
  S.Put(sSavingFile, 'Saving file #s.');
  ResFile.Put(@S,'Strings');
```

```
    S.Done;
    ResFile.Done;
  end;
```

Методы

Init constructor Init(AStrSize, AIndexSize: Word);

Создает в памяти список строк размера AStrSize с индексом из AIndexSize элементов. Буфер строк и буфер индексов заданного размера распределяются в куче. AStrSize должен быть достаточно велик для хранения всех строк, добавляемых в список строк — каждая строка занимает свою длину плюс 1 байт.

При добавлении строк в список строк (используя TStrListMaker.Put) строится индекс строк. Строки с последовательными ключами (такими как sInformation, sWarning и sError в предыдущем примере) записываются в одну индексную запись до 16. AIndexSize должен быть достаточно большим для добавления всех сгенерированных индексных записей. Каждый элемент индекса занимает 6 байт.

См. также: TStringList.Load, TStrListMaker.Done

Done destructor Done; virtual;

Освобождает память, распределенную этим объектом.

См. также: TStrListMaker.Init

Put procedure Put(Key: Word; S: String);

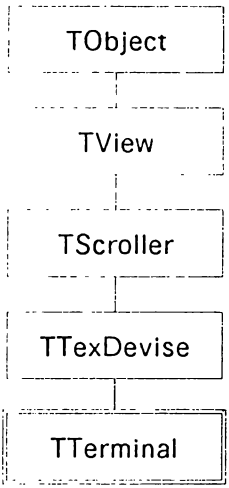
Добавляет String к списку строк (с заданным числовым Key).

Store procedure Store(var S: TStream);

Записывает список строк в поток.

TTerminal — TextView

TTerminal реализует «немой» терминал с буферизованным чтением и записью строк. По умолчанию — это циклический буфер размером 64K байт.



Поля

BufSize BufSize: Word; Только чтение

Размер буфера терминала в байтах.

Buffer Buffer: PTerminalBuffer; Только чтение

Указывает на первый байт буфера терминала.

QueFront QueFront: Word; Только чтение

Смещение (в байтах) первого байта запомненного в буфере терминала.

QueBack QueBack: Word; Только чтение

Смещение (в байтах) последнего байта, запомненного в буфере терминала.

Методы

Init constructor Init(var Bounds: TRect; AHScrollBar, AVScrollBar: PScrollBar; ABufSize: Word);

Перекрывается: Иногда

Создает объект TTerminal с данным Bounds, горизонтальной и вертикальной полосами скроллинга и буфером, вызывая TTextDevice.Init с аргументами Bounds и скроллерами, затем создает буфер (указываемый через Buffer) с BufSize равным ABufSize. GrowMode устанавливается в gfGrowHiX +

gfGrowHiY. QueFront и QueBack инициализируются в 0, указывая на пустой буфер. Курсор выводится в начале видимого элемента (0, 0).
См. также: TScroller.Init

Done destructor Done; virtual; Перекрывается: Иногда
Освобождает буфер и вызывает TTextDevice.Done для освобождения объекта.
См. также: TScroller.Done, TTextDevice.Done

BufDec procedure BufDec(var Val: Word);
Используется для манипуляции смещением очереди с кольцевым переносом: если Val = 0, Val устанавливается в BufSize–1; иначе Val уменьшается.
См. также: TTerminal.BufInc

BufInc procedure BufInc(var Val: Word);
Используется для манипуляции смещением очереди с кольцевым переносом: увеличивает Val на 1, если Val = BufSize, Val устанавливается в 0.
См. также: TTerminal.BufDec

CalcWidth function CalcWidth: Integer;
Возвращает длину самой длинной строки в текстовом буфере.

CanInsert function CanInsert(Amount: Word): Boolean;
Возвращает True, если число байт, заданное в Amount, можно вставить в буфер без уничтожения верхней строки.

Draw procedure Draw; virtual; Перекрывается: Редко
Вызывается когда скроллер TTerminal должен быть перерисован. Например, когда отмечена полоса скроллинга, когда изменен размер видимого элемента, когда изменены значения Delta или когда добавление текста приводит к скроллингу.

NextLine function NextLine(Pos: Word): Word;
Возвращает смещение в буфере начала строки, которое следует за позицией Pos.
См. также: TTerminal.PrevLines

PrevLines function PrevLines(Pos: Word; Lines: Word): Word;

Перекрывается: Иногда

Возвращает смещение в буфере начала строки, которое стоит перед позицией Pos.

StrRead function StrRead(var S: TextBuf) : Byte; virtual;

Перекрывается: Иногда

Абстрактный метод, возвращающий 0. Вы должны перекрыть его, если хотите, чтобы порожденный тип мог читать строки из текстового буфера.

StrWrite procedure StrWrite(var S:TextBuf; Count: Byte); virtual;

Перекрывается: Редко

Вставляет Count строк текста, заданного в S, в терминальный буфер. Этот метод управляет любым требуемым скроллингом, перерисовывая видимый элемент с помощью DrawView.

См. также: TView.DrawView

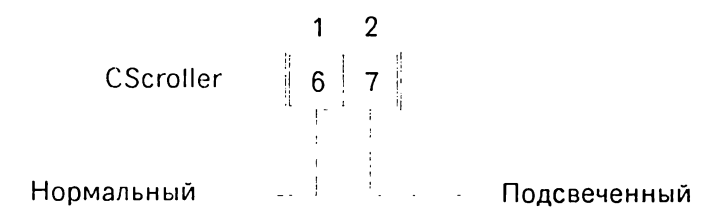
QueEmpty function QueEmpty: Boolean;

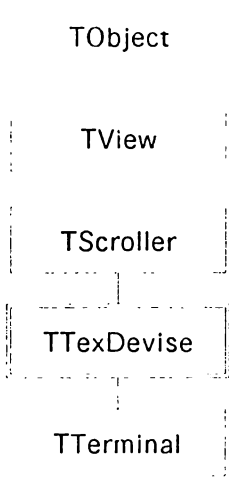
Возвращает True, если QueFront равна QueBack.

См. также: TTerminal,QueFront, TTerminal.QueBack

Палитра

Терминальные объекты используют палитру по умолчанию CScroller для отображения элементов 6 и 7 в палитру стандартной программы.





TTextDevice — это скроллингуемый драйвер устройства просмотра текста телетайпного типа. Кроме полей и методов, наследуемых от TScroller, TTextDevice определяет виртуальные методы для чтения и записи строк в и из устройства. TTextDevice существует как базовый тип для порождения реальных терминальных драйверов. TTextDevice использует конструктор и деструктор TScroller.

Методы

StrRead function StrRead(var S:TextBuf): Byte; virtual;

Перекрывается: Часто

Абстрактный метод, возвращающий по умолчанию 0. Вы должны перекрыть его в любом порожденном типе для чтения строки текстового устройства в S. Этот метод возвращает число прочитанных строк.

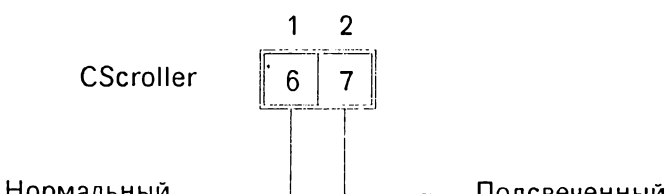
StrWrite procedure StrWrite(var S: TextBuf; Count: Byte); virtual;

Перекрывается: Всегда

Абстрактный метод для записи строки на устройство. Он должен перекрываться в порожденных типах. Например, TTerminal.StrWrite вставляет Count строк текста, заданных в S в терминальный буфер и перерисовывает видимый элемент.

Палитра

Объекты текстового устройства используют палитру по умолчанию CScroller для отображения в 6 и 7 элементы палитры стандартной программы.



TView — Views

Включите оператор

```
uses Views;
```

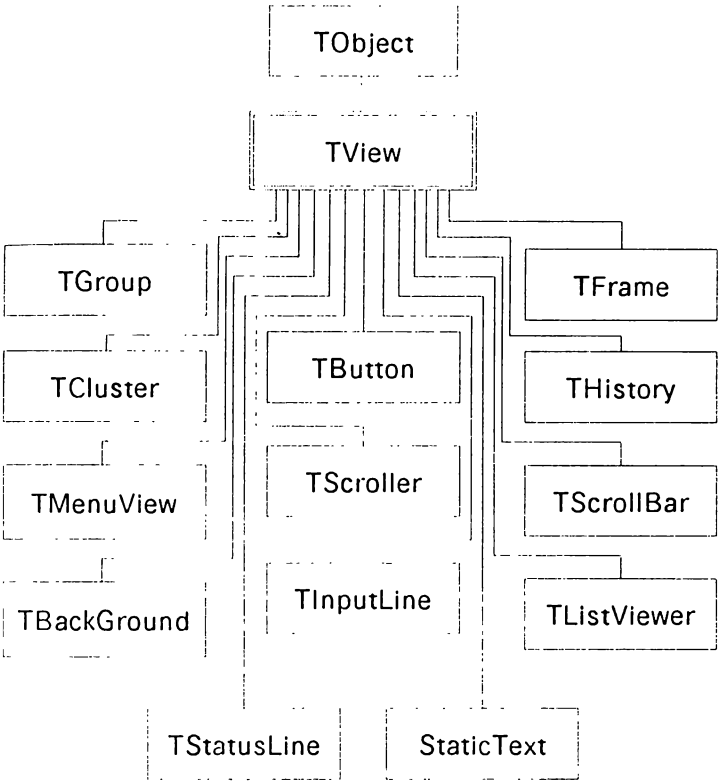
в программы, которые используют объекты TView, TFrame, TScrollBar, TScroller, TListViewer, TGroup, TWindow. Трудно представить программу на Turbo Vision, которая не использует эти объекты.

В программах на Turbo Vision редко создаются экземпляры объектов TView. Тип объекта TView обеспечивает основные поля и методы своим потомкам.

Поля

Owner `Owner: PGroup;` Только чтение

Указывает на объект TGroup, который владеет этим видимым элементом. Если nil, видимый элемент не имеет владельца. Видимый элемент ото-



бражается внутри видимого элемента владельца и будет отсекаться по прямоугольнику владельца.

Next — Next: PView; — Только чтение

Указывает на следующий равный видимый элемент в Z-порядке. Если это последний подэлемент, Next указывает на Owner первого подэлемента.

Origin — Origin: TPoint; — Только чтение

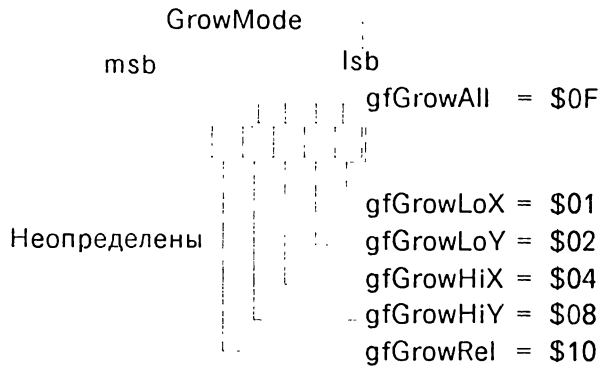
Координаты (X, Y) верхнего левого угла видимого элемента относительно Origin владельца.

См. также: MovTo, Locate

Size — Size: TPoint; — Только чтение

Размер видимого элемента.

Рис. 3.1. Отображение бит в GrowMode.



См. также: GrowTo, Locate

Cursor CurSor: TPoint; Только чтение

Положение аппаратного курсора внутри видимого элемента. Курсор видим только если видимый элемент активен (sfFocused) и курсор включен (sfCursorVis). Форма курсора или подчеркивание или блок (определен sfCursorIns).

См. также: SetCursor, ShowCursor, HideCursor, NormalCursor, BlockCursor

GrowMode GrowMode: Byte; Чтение/Запись

Определяет, как видимый элемент будет увеличиваться, когда его владелец изменяет размер. GrowMode определяется одной или более масками:

Пример: GrowMode := gfGrowLoX or gfGrowLoY;

См. также: gfXXXX константа grow mode

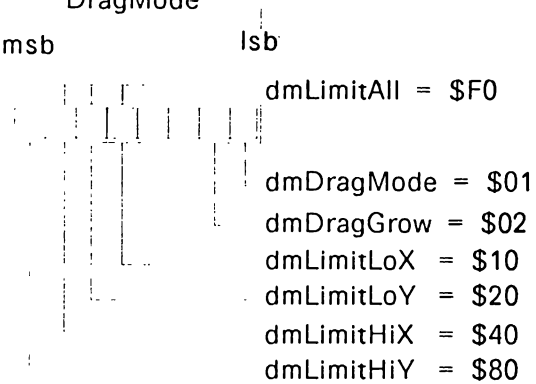
DragMode DragMode: Byte; Чтение/Запись

Определяет поведение видимого элемента при перемещении мышкой.

Биты DragMode определены, как показано на рис. 3.2.

Маски DragMode определены в «Константы DragMode dmXXXX» главы 4.

Рис. 3.2. Отображение бит в DragMode



См. также: TView.DragView

HelpCtx HelpCtx: Word; Чтение/Запись

Контекст подсказки видимого элемента. Когда видимый элемент активен, это поле представляет контекст подсказки программы, если номер контекста подсказки не hcNoContext (в этом случае нет контекста подсказки).

См. также: TView.GetHelpCtx

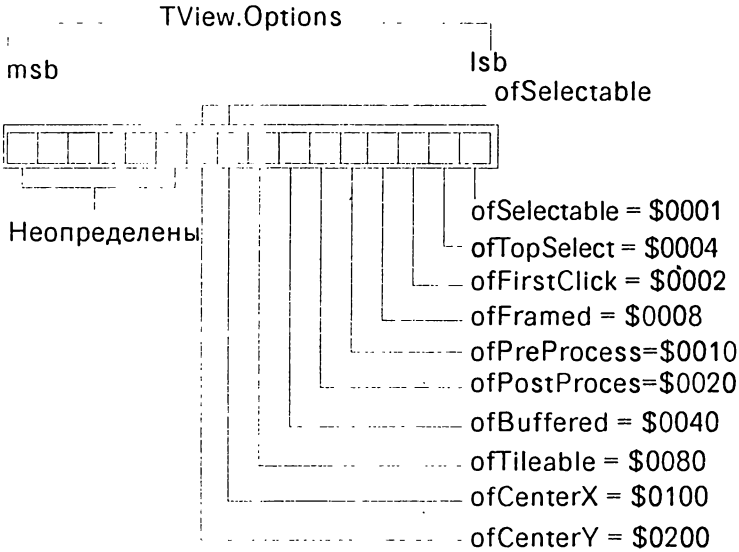
State State: Word; Только чтение

Состояние видимого элемента представлено набором бит в поле State. Многие методы TView проверяют и/или изменяют поле State, вызывая TView.SetState. TView.GetState (AState) возвращает True, если State видимого элемента равна AState. Биты State мнемонически представлены константами, описанными в «Константы флагов состояния sfXXXX» главы 4.

Options Options: Word; Чтение/Запись

Флаги Options определяют поведение видимого элемента. Биты Options определены:

Рис. 3.3. Биты в Options



Для детального описания флагов см. «Константы флагов опций ofXXXX» главы 4.

EventMask EventMask: Word; Чтение/Запись

EventMask — это битовая маска, которая определяет, какие классы событий будут распознаваться видимым элементом. По умолчанию EventMask разрешает evMouseDown, evKeyDown и evCommand. Назначение EventMask равным $FFFF заставляет видимый элемент реагировать на все классы событий; а значение 0 приводит к тому, что видимый элемент не реагирует на любые события. Для детального описания классов событий см. «Константы событий evXXXX» главы 4.

См. также: HandleEvent методы

Методы

Init constructor Init(var Bounds: TRect);

Перекрывается: Часто

Создает объект TView с прямоугольником Bounds. Init вызывает TObject.Init и создает поля нового TView со значениями:

Owner	nil
Next	nil
Origin	(Bounds.A.X, Bounds.A.Y)
Size	(Bounds.B.X – Bounds.A.X, Bounds.B.Y – Bounds.A.Y)
Cursor	(0, 0)
GrowMode	0
DragMode	dmLimitLoY
HelpCtx	hcNoContext
State	sfVisible
Options	0
EventMask	evMouseDown + evKeyDown + evCommand

Заметим, что TObject.Init заполняет нулями все поля потомков TView. Всегда вызывайте TView.Init до инициализации любых полей.

См. также: TObject.Init

Load constructor Load(var S: TStream);

Перекрывается: Часто

Создает объект TView и загружает его из потока S. Размер данных, читаемых из потока должен точно соответствовать размеру данных, записанных в поток методом Store. Если видимый элемент содержит указатели на равные видимые элементы, Load должен использовать GetPeerViewPtr для чтения этих указателей. Перекрытый конструктор Load всегда должен вызывать конструктор Load своего предка. По умолчанию TView.Load устанавливает поля Owner и Next в nil и читает оставшиеся поля из потока.

См. также: TView.Store, TStream.Get, TStream.Put

Done destructor Done; virtual; перекрывается: Часто
Скрывает видимый элемент и затем, если он имеет владельца, удаляет его из группы.

HandleEvent procedure HandleEvent(var Event: TEvent);
virtual;

Перекрывается: Всегда
Центральный метод, через который реализована вся обработка событий Turbo Vision. Поле What параметра Event содержит класс события (evXXXX) и оставшиеся поля Event описывают событие. Для указания, что он обработал событие, HandleEvent должен вызывать ClearEvent. HandleEvent почти всегда перекрывается в порожденных объектных типах. TView.HandleEvent обрабатывает события evMouseDown следующим образом: если видимый элемент не выбран (sfSelected) и не запрещен (sfDisabled) и если видимый элемент — выбираемый (ofSelectable), то видимый элемент выбирает себя, вызывая Select. Другие события в TView.HandleEvent не обрабатываются.
См. также: TView.ClearEvent

BlockCursor procedure BlockCursor;

Перекрывается: Никогда
Устанавливает sfCursorIns, чтобы изменить курсор в форму блока. Курсор будет видимым только если установлен sfCursorVis (и видимый элемент также видим).
См. также: sfCursorIns, sfCursorVis,
TView.NormalCursor, TView.ShowCursor,
TView.HideCursor

CalcBounds procedure CalcBounds(var Bounds: TRect;
Delta: TPoint); virtual;

Перекрывается: Редко
Когда владелец видимого элемента изменяет размер, он вызывает CalcBounds и ChangeBounds для всех подэлементов. CalcBounds должен вычислять новые границы видимого элемента на основе Delta,

на которое изменился размер его владельца, и возвращает новые границы в Bounds. TView.CalcBounds вычисляет новые границы, используя флаги, заданные в поле TView.GrowMode.
См. также: TView.GetBounds, TView.ChangeBounds, gfXXXX константы grow mode

ChangeBounds procedure ChangeBounds(var Bounds: TRect); virtual;

Перекрывается: Редко

Должен изменять границы видимого элемента (поля Origin и Size) на прямоугольник, заданный параметром Bounds. После изменения границ ChangeBounds должен перерисовать видимый элемент. ChangeBounds вызывается различными методами TView, но не должен вызываться прямо.
См. также: TView.Locate, TView.MoveTo, TView.GrowTo

ClearEvent procedure ClearEvent(var Event: TEvent);

Стандартный метод используется в HandleEvent для указания, что видимый элемент успешно обработал событие. Устанавливает Event.What в evNothing и Event.InfoPtr в @Self.
См. также: HandleEvent методы

CommandEnabled function CommandEnabled(Command: Word): Boolean;

Возвращает True, если команда Command разрешена, иначе — False. Заметим, что когда вы изменяете модальное состояние, вы можете запретить или разрешить необходимые команды; однако, когда вы возвращаетесь в предыдущее модальное состояние, оригинальный набор команд должен быть восстановлен.
См. также: TView.DisableCommand, TView.EnableCommand, TView.SetCommands

DataSize function DataSize: Word; virtual;

Перекрывается: Редко

DataSize должен возвращать размер данных читаемый из и записываемый в запись данных с помощью SetData и GetData. Механизм записи данных обычно используется только в видимых элементах, которые реализуют элементы управления для диалоговых окон.

См. также: TView.GetData, TView.SetData

DisableCommands procedure DisableCommands (Commands: TCommandSet);

Запрещает команды, указанные в аргументе Commands.

См. также: TView.CommandEnabled, TView.EnableCommands, TView.SetCommands

DragView procedure DragView(Event: TEvent; Mode: Byte; var Limits: TRect; MinSize, MaxSize: TPoint);

Перемещает мышкой видимый элемент, используя режим перемещения, заданный флагами dmXXXX в Mode. Limits указывает прямоугольник (в координатной системе владельца) внутри которого может перемещаться видимый элемент, а Min и Max указывают минимальный и максимальный размеры видимого элемента, которые он может принимать. Событие, приводящее к операции перемещения мышкой, требуется в Event для различения перемещения мышкой от использования клавиш курсора.

См. также: TView.DragMode, dmXXXX константы drag mode

Draw procedure Draw; virtual; Перекрывается: Всегда

Вызывается, когда видимый элемент должен перерисовать себя. Draw должен покрывать всю область видимого элемента. Этот метод должен соответственно перекрываться для каждого порожденного объекта. Draw редко вызывается прямо, поскольку более эффективно использовать DrawView, который рисует только ожидаемые видимые элементы, т.е. когда часть или весь видимый элемент видим на экране. При необходимости Draw может вызвать GetClipRect для получения прямоугольника, кото-

рый необходимо перерисовать и затем рисовать только эту область. Для сложных видимых элементов это может значительно улучшить производительность.

См. также: TView.DrawView

DrawView procedure DrawView;

Вызывает Draw, если TView.Exposed возвращает True, указывая, что видимый элемент ожидается (см. sfExposed). Вы должны вызвать DrawView (а не Draw), когда вам необходимо перерисовать видимый элемент после изменения, которое воздействует на его видимое отображение.

См. также: TView.Draw, TGroup.ReDraw, TView.Exposed

EnableCommands procedure EnableCommands(Commands: TCommandSet);

Разрешает все команды в аргументе Commands.

См. также: TView.DisableCommands, TView.GetCommands, TView.CommandEnabled, TView.SetCommands

EndModal procedure EndModal(Command: Word); virtual;

Перекрывается: Никогда

Завершает текущее модальное состояние и возвращает Command, как результат вызова функции ExecView, которая создала это модальное состояние.

См. также: TGroup.ExecView, TGroup.Execute, TGroup.EndModal

EventAvail function EventAvail: Boolean;

Возвращает True, если событие доступно для GetEvent.

См. также: TView.MouseEvent, TView.KeyEvent, TView.GetEvent

Execute function Execute: Word; virtual;

Перекрывается: Редко

Вызывается из TGroup.ExecView когда видимый элемент становится модальным. Если видимый эле-

мент разрешает модальное выполнение, он должен перекрыть Execute для выполнения цикла событий. Результатом Execute становится значение, возвращаемое из TGroup.ExecView.
TView.ExecView просто возвращает cmCancel.
См. также: sfModal, TGroup.Execute, TGroup.ExecView

Exposed function Exposed: Boolean;
Возвращает True, если любая часть видимого элемента видна на экране.
См. также: sfExposed, TView.DrawView

GetBounds procedure GetBounds(var Bounds: TRect);
Возвращает в переменной Bounds граничный прямоугольник видимого элемента в координатной системе его владельца. Bounds.A устанавливается в Origin, а Bounds.B устанавливается в Origin + Size.
См. также: TView. Origin, TView.Size, TView.CalcBounds, TView.ChangeBounds, TView.SetBounds, TView.GetExtent

GetClipRect procedure GetClipRect(var Clip: TRect);
Возвращает в переменной Clip минимальный прямоугольник, который требуется перерисовать в вызове Draw. Для сложных видимых элементов Draw может использовать GetClipRect для значительного улучшения производительности.
См. также: TView.Draw

GetColor function GetColor(Color: Word): Word;
Отображает индексы палитры младшего и старшего байта Color в физические атрибуты символа, проходя через палитру видимого элемента и палитры всех его владельцев.
См. также: TView.GetPalette

GetCommands procedure GetCommands(var Commands: TCommandSet);
Возвращает в Commands текущий набор команд.

См. также: TView.CommandsEnabled,
TView.EnableCommands,
TView.DisableCommands, TView.SetCommands

GetData procedure GetData(var Rec); virtual;

Перекрывается: Редко

Должна копировать DataSize байт из видимого элемента в запись данных Rec. Механизм записей данных обычно используется только в видимых элементах, которые реализуют элементы управления для диалоговых окон. По умолчанию TView.GetData ничего не делает.

См. также: TView.DataSize, TView.SetData

GetEvent procedure GetEvent(var Event: TEvent); virtual;

Перекрывается: Редко

Возвращает следующее доступное событие в TEvent. Возвращает evNothing, если событие недоступно. По умолчанию он вызывает GetEvent владельца видимого элемента.

См. также: TView.EventAvail, TProgram.Idle,
TView.HandleEvent, TView.PutEvent

GetExtent procedure GetExtent(var Extent: TRect);

Возвращает в переменной Extent прямоугольник видимого элемента. Extent.A устанавливается в (0,0), а Extent.B устанавливается в Size.

См. также: TView.Origin, TView.Size,
TView.CalcBounds, TView.ChangeBounds,
TView.SetBounds, TView.GetBounds

GetHelpCtx function GetHelpCtx: Word; virtual;

Перекрывается: Редко

Должна возвращать контекст подсказки видимого элемента. По умолчанию TView.GetHelpCtx возвращает значение поля HelpCtx или возвращает hcDragging, если видимый элемент перемещается мышкой (см. sfDragging).

См. также: HelpCtx

GetPalette function GetPalette: PPalette; virtual;

Перекрывается: Всегда

Должна возвращать указатель на палитру видимого элемента или nil, если видимый элемент не имеет палитры. GetPalette вызывается из GetColor, WriteChar и WriteStr при преобразовании индексов палитры в физические атрибуты символов. При возвращаемом значении nil трансляция цвета этим видимым элементом не производится. GetPalette почти всегда перекрывается в порожденных типах объектов.

См. также: TView.GetColor, TView.WriteXXXX

GetPeerViewPtr procedure GetPeerViewPtr(var S: TStream; var P);

Загружает указатель P равного видимого элемента из потока S. Равный видимый элемент — это видимый элемент с тем же владельцем, что и у этого видимого элемента; например TScroller содержит два равных видимых элемента HScrollBar и VScrollBar. GetPeerViewPtr должен использоваться только внутри конструктора Load для чтения значений указателей, которые были записаны при вызове PutPeerViewPtr из метода Store. Загруженное в P значение не является действительным до тех пор, пока владелец видимого элемента не завершит полностью операцию Load; следовательно, ссылка по указателю на равный видимый элемент внутри конструктора Load не дает корректного значения.

См. также: TView.PutPeerViewPtr, TGroup.Load, TGroup.Store

GetState function GetState(AState: Word): Boolean;

Возвращает True, если состояние в AState установлено в поле State.

См. также: State, TView.SetState

GrowTo procedure GrowTo(X, Y: Integer);

Увеличивает или уменьшает видимый элемент на данный размер, используя вызов TView.Locate.

См. также: TView.Origin, TView.Locate, TView.MoveTo

Hide procedure Hide;

Прячет видимый элемент, вызывая SetState для очистки флага sf Visible в State.

См. также: sfVisible, TView.SetState, TView.Show

HideCursor procedure HideCursor;

Прячет курсор, очищая бит sfCursorVis в State.

См. также: sfCursorVis, TView.ShowCursor

KeyEvent procedure KeyEvent(var Event: TEvent);

Возвращает в переменной Event следующее событие evKeyDown. Он ожидает, игнорируя все другие события, до тех пор, пока событие от клавиатуры не будет доступно.

См. также: TView.GetEvent, TView.EventAvail

Locate procedure Locate(var Bounds: TRect);

Изменяет границы видимого элемента на Bounds. Видимый элемент перерисовывается в его новом положении. Locate вызывает SizeLimits для проверки, что данные Bounds правильны, затем вызывает ChangeBounds для изменения границ и перерисовывает видимый элемент.

См. также: TView.GrowTo, TView.MoveTo, TView.ChangeBounds

MakeFirst procedure MakeFirst;

Перемещает видимый элемент на вершину списка подэлементов владельца. Вызов MakeFirst соответствует PutInFrontOf(Owner^.First).

См. также: TView.PutInFrontOf

MakeGlobal procedure MakeGlobal(Source: TPoint;
var Dest:TPoint);

Преобразует координаты в точке Source из локальных (видимый элемент) в глобальные (экран) и возвращает результат в Dest. Source и Dest могут быть одной переменной.

См. также: TView.MakeGlobal, TView.MouseInView

MakeLocal procedure MakeLocal(Source: TPoint;
var Dest:TPoint);

Преобразует координаты точки Source из глобальных (экран) в локальные (видимый элемент) и возвращает результат в Dest. Полезно для преобразования поля Event.Where в событии evMouse из глобальных координат в локальные. Например MakeLocal(Event.Where, MouseLoc).

См. также: TView.MakeGlobal, TView.MouseInView

MouseEvent function MouseEvent(var Event: TEvent; Mask: Word): Boolean;

Возвращает следующее событие от мышки в Event. Возвращает True, если возвращенное событие есть в аргументе Mask, и False, если возникло событие evMouseUp. Этот метод позволяет вам трассировать мышку, когда ее кнопка нажата, например, в операциях отметки блока в текстовых редакторах.

Приведем фрагмент программы HandleEvent, которая следит за мышкой с курсором видимого элемента.

```
procedure TMyView.HandleEvent(var
                                    Event: TEvent);
begin
  TView.HandleEvent(Event);
  case Event.What of
    evMouseDown:
      begin
        repeat
          MakeLocal(Event.Where, Mouse);
          SetCursor(Mouse.X, Mouse.Y);
        until not MouseEvent(Event,
            evMouseMove);
        ClearEvent(Event);
      end;
      ...
    end;
end;
```

См. также: EventMasks, TView.KeyEvent, TView.GetEvent

MouseInView function MouseInView(Mouse: TPoint): Boolean;

Возвращает True, если аргумент Mouse (заданный в глобальных координатах) внутри видимого элемента.

См. также: TView.MakeLocal

MoveTo procedure MoveTo(X, Y: Integer);

Перемещает Origin в точку (X, Y) относительно владельца видимого элемента. Size видимого элемента не изменяется.

См. также: Origin, Size, TView.Locate, TView.GrowTo

NextView function NextView: PView;

Возвращает указатель на следующий подэлемент в списке подэлементов владельца. Возвращается nil, если видимый элемент последний в списке владельца.

См. также: TView.PRevView, TView.Prev, TView.Next

NormalCursor procedure NormalCursor;

Очищает бит sfCursorIns в State, переводя курсор в режим подчеркивания. Если sfCursorVis установлен, новый курсор будет отображаться.

См. также: sfCursorIns, sfCursorVis, TView.HideCursor, TView.BlockCursor, TView.HideCursor

Prev function Prev: PView;

Возвращает указатель на предыдущий подэлемент в списке подэлементов владельца. Если видимый элемент первый в списке владельца, Prev возвращает последний видимый элемент в списке. Заметим, что TView.Prev интерпретирует список как кольцевой, в то время как TView.PrevView интерпретирует его как линейный.

См. также: TView.NextView, TView.PrevView, TView.Next

PrevView function PrevView: PView;

Возвращает указатель на предыдущий подэлемент в списке подэлементов владельца. Возвращается nil, если видимый элемент — первый в списке владельца. Заметим, что TView.Prev интерпретирует список как кольцевой, а TView.PrevView — как линейный.

См. также: TView.NextView, TView.Prev

PutEvent procedure PutEvent(var Event: TEvent); virtual;

Перекрывается: Редко

Помещает событие, заданное в Event, в очередь событий, в результате чего это событие будет следующим событием, возвращаемым GetEvent. Этим способом в очередь может быть помещено только одно событие. Это часто используется видимыми элементами, генерирующими командные события, например:

```
Event.What := evCommand;
Event. Command := cmSaveAll;
Event.InfoPtr := nil;
PutEvent(Event);
```

По умолчанию TView.PutEvent вызывает PutEvent владельца видимого элемента.

См. также: TView.EventAvail, TView.GetEvent, TView.HandleEvent

PutInFontOf procedure PutInFontOf(Target: PView);

Помещает видимый элемент перед видимым элементом Target в списке подэлементов владельца. Вызов

```
TView.PutInFontOf(Owner^.First);
```

эквивалентен TView.MakeFirst. Этот метод работает, изменяя указатели в списке подэлементов. В зависимости от позиции других видимых элементов и их состояния, PutInFrontOf может отсекать

закрываемые видимые элементы. Если видимый элемент — выбираемый (см. ofSelectable) и помещается сверху всех других подэлементов, этот видимый подэлемент становится выбранным.

См. также: TView.MakeFirst

PutPeerViewPtr procedure PutPeerVieewPtr(var S: TStream; P: PView);

Сохраняет указатель P на равный видимый элемент в потоке S. Равный видимый элемент — это видимый элемент с тем же владельцем, что и этот видимый элемент. PutPeerViewPtr должен использоваться только внутри метода Store для записи значений указателей, которые позже могут быть считаны с помощью GetPeerViewPtr в конструкторе Load.

См. также: TView.PutPeerViewPtr, TGroup.Load, TGroup.Store

Select procedure Select;

Выбирает видимый элемент (см. sfSelected). Если владелец видимого элемента активизируется, то видимый элемент также активизируется (см. sfFocused). Если видимый элемент имеет установленный флаг ofTopSelect в поле Options, то видимый элемент перемещается на вершину списка подэлементов владельца (используя вызов TView.MakeFirst).

См. также: sfSelected, sfFocused, ofTopSelect, TView.MakeFirst

SetBounds procedure SetBounds(var Bounds: TRect);

Устанавливает граничный прямоугольник видимого элемента в значения параметра Bounds. Поле Origin устанавливается в Bounds.A и поле Size устанавливается в Bounds.B — Bounds.A. Метод SetBounds вызывается только из перекрытого метода ChangeBounds — вы не должны вызывать SetBounds прямо.

См. также: TView.Origin, TView.Size, TView.CalcBounds, TView.ChangeBounds, TView.GetBounds, TView.GetExtent

SetCommands procedure SetCommands(Commands: CommandSet);

Изменяет текущий набор команд на заданный аргументом Commands.

См. также: TView.EnableCommands, TView.DisableCommands

SetCursor procedure SetCursor(X, Y: Integer);

Перемещает аппаратный курсор в точку (X,Y) используя относительные координаты видимого элемента (локальные). (0,0) — это верхний левый угол.

См. также: TView.MakeLocal, TView.HideCursor, TView.ShowCursor

SetData procedure SetData(var Rec); virtual;

Перекрывается: Редко

SetData должен копировать DataSize байт из записи данных Rec в видимый элемент. Механизм записи данных обычно используется только в видимых элементах, которые реализуют элементы управления для диалоговых окон.

По умолчанию TView.SetData ничего не делает.

См. также: TView.DataSize, TView.GetData

SetState procedure SetState(AState: Word; Enable: Boolean); virtual;

Перекрывается: Иногда

Устанавливает или очищает флаг состояния в поле TView.State. Параметр AState задает флаг состояния для модификации (см. sfXXXX), а параметр Enable указывает, будет этот флаг устанавливаться (True) или выключаться (False). TView.SetState затем выполняет соответствующие действия для отражения нового состояния, такие как перерисовка видимых элементов, которые появляются, когда данный видимый элемент скрывается (sfVisible) или перепрограммирование аппаратуры, когда из-

меняется форма курсора (sfCursorVis и sfCursorIns).

SetState иногда перекрывается для выполнения дополнительных действий, основанных на флагах состояний. Например, тип TFrame перекрывает SetState для своей перерисовки, когда окно выбирается или перемещается мышкой.

```
procedure TFrame.SetState(AState: Word;
                          Enable: Boolean);
begin
  TView.SetState(AState, Enable);
  if AState and (sfActive + sfDragging) <> 0 then
    DrawView;
end;
```

Другая причина перекрыть SetState — это разрешить или запретить команды, которые обрабатываются определенным видимым элементом.

```
procedure TMyView.SetState(AState: Word;
                           Enable: Boolean);
const
  MyCommands = [cmOut, cmCopy, cmPaste,
                cmClear];
begin
  TView.SetState(AState, Enable);
  if AState = sfSelected then
    if Enable then
      EnableCommands(MyCommands) else
      DisableCommands(MyCommands);
end;
```

См. также: TView.GetState, TView.State, sfXXXX константы state flag

Show procedure Show;

Показывает видимый элемент, вызывая SetState для установки флага sfVisible в поле State.

См. также: TView.SetState

ShowCursor procedure ShowCursor;

Включает аппаратный курсор, устанавливая sfCursorVis. Заметим, что по умолчанию курсор невидим.

См. также: sfCursorVis, TView.HideCursor

SizeLimits procedure SizeLimits(var Min, Max: TPoint); virtual;

Перекрывается: Иногда

Возвращает в переменных Min и Max, минимальное и максимальное значения, которые может принимать поле Size.

См. также: TView.Size

Store procedure Store(var S: TStream);

Перекрывается: Часто

Сохраняет видимый элемент в потоке S. Размер данных, записываемых в поток, должен точно соответствовать размеру данных, читаемых из потока конструктором Load видимого элемента. Если видимый элемент содержит указатель на равные видимые элементы, Store должен использовать PutPeerViewPtr для записи этих указателей. Перекрытый метод Store должен всегда вызывать родительский метод Store. По умолчанию TView.Store пишет все поля, кроме Owner и Next в поток.

См. также: TView.Load, TStream.Get, TStream.Put

TopView function TopView: PView;

Возвращает указатель на текущий модальный видимый элемент.

Valid function Valid(Commands: Word): Boolean; virtual;

Перекрывается: Иногда

Этот метод используется для проверки правильности видимого элемента после его создания (с использованием Init или Load) или в момент, когда заканчивается модальное состояние (при вызове EndModal).

Значение параметра Command, равное cmValid (ноль), указывает, что видимый элемент должен проверять результат своего создания:

Valid(cmValid) должен возвращать True, если видимый элемент был успешно создан и готов к использованию; иначе — False.

Любое другое (не ноль) значение параметра Command указывает, что текущее модальное состояние (такое как модальное диалоговое окно) должно завершаться с результирующим значением Command. В этом случае Valid должна проверять правильность видимого элемента.

Valid должна сообщить пользователю, если видимый элемент неправильный, например используя программу MessageBox в модуле StdDlg.

Типы объектов, определенные в модуле StdDlg содержат ряд примеров перекрытия Valid.

По умолчанию TView.Valid просто возвращает True.

См. также: TGroup.Valid, TDialog.Valid, TProgram.ValidView

WriteBuf procedure TView.WriteBuf(X, Y, W, H: Integer; var Buf);

Записывает буфер на экран, начиная с координат (X,Y) и заполняет область шириной W и высотой H. Должен использоваться только в методах Draw. Обычно параметр Buf типа TDrawBuffer, но может быть любым массивом слов, где каждое слово содержит символ в младшем байте и атрибут в старшем байте.

См. также: TView.Draw

WriteChar procedure TView.WriteChar(X, Y: Integer; Ch: Char; Color: Byte; Count: Integer);

Начиная с точки (X,Y) записывает Count копий символа Ch цветом, определенным элементом с номером Color, в палитре текущего видимого элемента. Должен использоваться только в методах Draw.

См. также: TViewdraw

WriteLine procedure TView.WriteLine(X, Y, W, H: Integer;
var Buf);

Записывает строку, содержащуюся в буфере Buf на экран, начиная с точки (X,Y) и внутри прямоугольника, определенного шириной W и высотой H. Если H больше 1, строка будет повторяться H раз. Должен использоваться только в методе Draw. Параметр Buf обычно типа TDrawBuffer, но может быть любым массивом слов, где каждое слово содержит символ в младшем байте и атрибут — в старшем байте.

См. также: TView.Draw

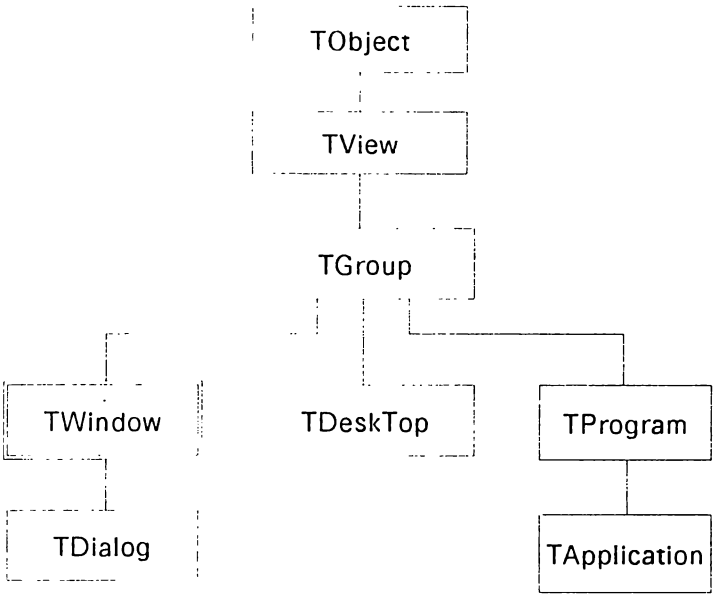
WriteStr procedure TView.WriteStr(X, Y: Integer;
Str: String; Color: Byte);

Записывает строку Str с цветом элемента с номером Color в палитре видимого элемента, начиная с точки (X,Y). Должен использоваться только в методе Draw.

См. также: TView.Draw

TWindow — Views

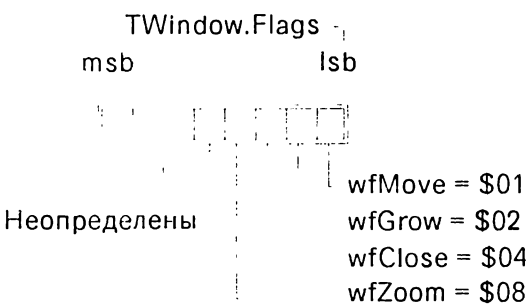
Объект TWindow — это группа, которая обычно владеет объектом TFrame, объектом интерьера TScroller и одним или двумя объектами TScrollBar. Объект TFrame задает обычную рамку, размещает необязательный заголовок и номер и функциональные кнопки (закрытия, масштабирования, перемещения). Объекты TWibdow имеют встроенные возможности перемещения и масштабирования с помощью мышки и клавиатуры. Они могут масштабироваться и закрываться отметкой мышки соответствующих кнопок. Они также «знают», как работать с полосами скроллинга и скроллера. Окна с номерами от 1 до 9 могут выбираться клавишами Alt-n (n от 1 до 9).



Поля

Flags Flags: Byte; Чтение/Запись

Поле Flags содержит комбинацию следующих бит:



Описание флагов окна см. в «Константы флагов окна wfXXXX» главы 4.

ZoomRect ZoomRect: TRect; Только чтение

Нормальные немасштабированные границы окна.

Number Number: Integer; Чтение/Запись

Номер этого окна. Если TWindow.Number в диапазоне 1–9, номер будет появляться в заголовке рамки и окно может быть выбрано клавишами Alt-n (n от 1 до 9).

Palette Palette: Integer; Чтение/Запись

Определяет какая палитра окна будет использоваться: wpBlueWindow, wpCyanWindow, wpGrayWindow. Палитра по умолчанию wpBlueWindow.

См. также: TWindow.GetPalette, wpXXXX константы

Frame Frame: PFrame; Только чтение

Frame — это указатель на объект TFrame, связанный с окном.

См. также: TWindow.InitFrame

Title Title: RString; Чтение/Запись

Строка символов, определяющая (необязательный) заголовок, который появляется в рамке.

Методы

Init constructor Init(var Bounds: TRect; ATitle: TTitleStr; ANumber: Integer);

Вызывает TGroup.Init(Bounds). Устанавливает State в sfShadow. Устанавливает по умолчанию Options в (ofSelectable + ofTopSelect). Устанавливает GrowMode в gfGrowAll + gfGrowRel. Устанавливает Flags в (wfMove + wfGrow + wfClose + wfZoom). Устанавливает поле Title в NewStr(ATitle), поле Number в ANumber. Вызывает InitFrame, и если поле Frame не nil, вставляет его в группу окна. Наконец, устанавливает ZoomRect в Bounds.

См. также: TFrame.InitFrame

Load constructor Load(var S: TStream);

Создает и загружает окно из потока S, вызывая TGroup.Load, затем читая дополнительные поля, введенные в TWindow.

См. также: TGroup.Load

Done destructor Done; virtual; Перекрывается: Иногда

Освобождает окно и подэлементы.

Close procedure Close; virtual; Перекрывается: Редко

Закрывает и освобождает окно обычно в ответ на командное событие cmClose. Соответствует вызову деструктора Done.

GetPalette function GetPalette: PPalette; virtual;

Перекрывается: Иногда

Возвращает указатель на палитру, заданную индексом палитры в поле Palette.

См. также: TWindow.Palette

GetTitle function GetTitle(MaxSize: Integer): TTitleStr; virtual;

Перекрывается: Редко

Должна возвращать строку заголовка окна. Если строка заголовка больше, чем MaxSize символов, GetTitle должна пытаться сократить ее; иначе она будет отсекаться отбрасыванием текста после MaxSize символов. TFrame.Draw вызывает Owner^.GetTitle для получения строки текста, отображаемой в рамке.

По умолчанию GetWindow.Title возвращает строку Title^ или пустую строку, если поле Title равно nil.

См. также: TWindow.Title, TFrame.Draw

HandleEvent procedure HandleEvent(var Event: TEvent); virtual;

Перекрывается: Часто

Вначале вызывает TGroup.HandleEvent, затем обрабатывает специфические для TWindow события: события evCommand обрабатываются, если поле TWindow.Flags разрешает эту операцию: cmResize (переместить или изменить размер окна, используя

метод TView.DrawView), cmClose (закрыть окно, используя метод TWindow.Close) и cmZoom (масштабировать окно, используя метод TWindow.Zoom). События evKeyDown со значением KeyCode, равным kbTab или kbShiftTab обрабатываются, выбирая следующий или предыдущий выбираемый подэлемент (если он существует).

Событие evBroadcast со значением Command, равным cmSelectWindowNum, обрабатывается выбором окна, если поле Event.InfoInt равно TWindow.Number.

См. также: TGroup.HandleEvent, wfXXXX константы

InitFrame procedure InitFrame; virtual;

Перекрывается: Редко

Создает объект TFrame для окна и сохраняет указатель на эту рамку в поле TWindow.Frame. InitFrame вызывается из TWindow.Init и никогда не должен вызываться прямо. InitFrame может быть перекрыт для создания экземпляра объекта, порожденного от TFrame.

См. также: TWindow.Init

SetState procedure SetState(AState: Word;
Enable: Boolean); virtual;

Перекрывается: Редко

Вначале вызывает TGroup.SetState. Затем, если AState равно sfSelected, активизирует или деактивизирует окно и все его подэлементы, используя вызов SetState(sfActive, Enable) и вызов TView.EnebleCommands или TView.DisableCommands для cmNext, cmPrev, cmResize, cmClose и cmZoom.

См. также: TGroup.SetState, EnableCommands, DisableCommands

SizeLimits procedure SizeLimits(var Min, Max: TPoint); virtual;

Перекрывается: Редко

Перекрывает TView.SizeLimits. Вызывает TView.SizeLimits, затем изменяет Min на значение, хранимое в глобальной переменной MinWindowSize.

См. также: TView.SizeLimits, MinVinSize переменная

StandardScrollBar function StandardScrollBar(AOptions: Word): PScrollBar;

Создает, вставляет и возвращает указатель на «стандартную» полосу скроллинга для окна. «Стандартный» означает, что полоса скроллинга вставляется в рамку окна не закрывая углов или кнопки масштабирования.

Параметр AOptions может быть либо sbGorizontal для создания горизонтальной полосы скроллинга внизу окна, либо sbVertical для создания вертикальной полосы скроллинга в правой стороне окна, либо может быть скомбинировано с sbHandleKeyboard, чтобы разрешить полосе скроллинга откликаться на клавиши стрелок и страниц, а не только отметок от мышки.

См. также: sbXXXX константы scroll bar

Store procedure TWindow.Store(var S: TStream);

Сохраняет окно в потоке S, вызывая TGroup.Store, затем записывая дополнительные поля, определенные в TWindow.

См. также: TGroup.Store

Zoom procedure TWindow.Zoom; virtual;

Перекрывается: Редко

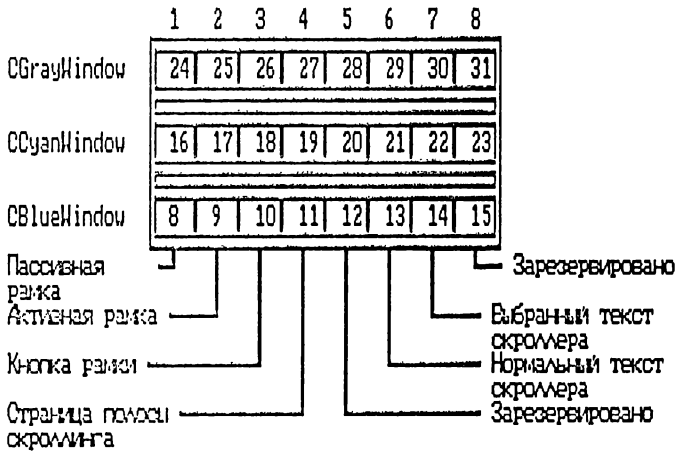
Масштабирует окно. Этот метод обычно вызывается в ответ на команду cmZoom (генерируемую при отметке кнопки масштабирования). Zoom принимает во внимание относительные размеры окна и его владельца, и значение ZoomRect.

См. также: cmZoom, ZoomRect

Палитра

Объекты окна используют палитры по умолчанию CBlueWindow (для текстовых окон), CCyanWindow

(для сообщений) и CGrayWindow (для диалоговых окон).



ГЛОБАЛЬНЫЕ ССЫЛКИ

Эта глава описывает все элементы Turbo Vision, которые не являются частью иерархии стандартных объектов Turbo Vision. Все стандартные объекты описаны в главе 3.

Элементы этой главы включают типы, константы, переменные, процедуры и функции, определенные в модулях Turbo Vision. Типичный элемент главы выглядит так:

Процедура Sample — Модуль

Объявление procedure Sample(AParameter);
Функция Sample выполняет ряд полезных функций с параметром, AParameter.
См. также Функция Example

Процедура Abstract — Objects

Объявление procedure Abstract;
Функция Вызов этой процедуры завершает программу с ошибкой времени выполнения 211. При реализации абстрактных типов объекта используйте вызовы Abstract в тех виртуальных методах, которые должны быть перекрыты в порожденных типах. Это предотвратит любые попытки использования экземпляров абстрактного типа.
См. также «Абстрактные методы» в главе 3 Описания.

Переменная Application — App

Объявление Application: PApplicaton = nil;
Функция Переменная Application устанавливается в @Self в начале TProgram.Init (вызывается в TApplication.Init) и очищается в nil в конце

TProgram.Done (вызывается в TApplication.Done). Поэтому, во время выполнения программы на Turbo Vision, Application указывает на объект-программу.

См. также TProgram.Init

Переменная AppPalette — App

Объявление AppPalette: Integer = apColor;

Функция Выбирает одну из трех доступных в программе палитр (apColor, apBlackWhite, apMonochrome). AppPalette инициализируется TProgram.InitScreen в зависимости от текущего режима экрана и используется TProgram.GetPalette для возврата палитры. Вы можете перекрыть TProgram.InitScreen, чтобы изменить выбор палитры по умолчанию.

См. также TProgram.InitScreen, константы apXXXX

Константы apXXXX — App

Значения Определены следующие константы палитры:

Таблица 4.1

Константы палитры

Константа	Значение	Назначение
apColor	0	Палитра для цветного монитора
apBlackWhite	1	Палитра для LCD монитора
apMonochrome	2	Палитра для монохромного монитора

Функция Константы, начинающиеся с ap, используются для указания, с какой из трех стандартных палитр будет работать программа на Turbo Vision. Три палитры используются для цветного, черно-белого и монохромного дисплеев.

Процедура AssignDevice — TextView

Объявление
```
procedure AssignDevice(var T: Text;
                       Screen: PTextDevice);
```

Функция Связывает текстовый файл с TTextDevice. AssignDevice работает аналогично стандартной процедуре Assign за исключением того, что указывается не имя файла. Вместо этого текстовый файл связывается с TTextDevice данным в Screen (запоминая Screen в первых 4 байтах поля UserData в TextRec(T). Последовательность операций ввода-вывода для текстового файла будет читать и писать из TTextDevice, используя виртуальные методы StrRead и StrWrite. Поскольку TTextDevice — это абстрактный тип, параметр Screen обычно указывает на образец TTerminal, который реализует полную функциональность видимого элемента телетайпно-подобного скроллинга.

См. также TTextDevice; TextRec

Константы bfXXXX — Dialogs

Значения Определены следующие флаги кнопки:

Таблица 4.2

Константа	Значение	Назначение
bfNormal	$00	Нормальная кнопка
bfDefault	$01	Кнопка по умолчанию
bfLeftJust	@02	Метка кнопки выровнена влево

Функция Комбинация этих значений передается в TButton.Init для определения вновь созданного стиля кнопки. bfNormal указывает на нормальную не умалчиваемую кнопку. bfDefault указывает что кнопка будет кнопкой по умолчанию. Обязанность программиста — обеспечить, чтобы кнопка была единственной умалчиваемой кнопкой в TGroup. Значение

bfLeftJust может быть добавлено к bfDefault или bfNormal и воздействует на позицию отображаемого текста внутри кнопки: если очищено, то метка центрируется; если установлено, то метка выравнивается влево.

См. также TButton.Flags, TButton.MakeDefault, TButton.Draw

Переменная ButtonCount — Drivers

Объявление ButtonCount: Byte = 0;

Функция ButtonCount хранит число кнопок мышки или 0, если мышка не инсталлирована. Вы можете использовать эту переменную для определения, доступна ли поддержка мышки. Значение устанавливается в инициализационном коде Drivers и не может быть изменено.

Переменная CheckSnow — Drivers

Объявление CheckSnow: Boolean;

Функция CheckSnow выполняет функцию одноименного флага стандартного модуля Turbo Pascal — Crt. Проверка «снега», замедляющая вывод на экран, требуется только для некоторых старых адаптеров CGA.

См. также InitVideo

Процедура ClearHistory — HistList

Объявление procedure ClearHistory;

Функция Удаляет все строки из всех списков истории.

Процедура ClearScreen — Drivers

Объявление procedure ClearScreen;

Функция Очищает экран. ClearScreen предполагает, что вначале был вызван InitVideo. Вам редко потребуется использовать эту процедуру, как это объяснено в описании InitVideo.

См. также InitVideo

Функция Эти константы представляют предопределенные команды Turbo Vision. Они передаются в поле TEvent.Command событий evMessage(evCommand и evBroadcast) и заставляют методы HandleEvent стандартных объектов Turbo Vision выполнять различные задачи.

Turbo Vision резервирует значения констант от 0 до 99 и от 256 до 999 для своих целей. Обработчики событий стандартных объектов Turbo Vision реагируют на эти предопределенные константы. Программисты могут определить свои собственные константы в диапазонах от 100 до 255 и от 1000 до 65535 без конфликтов с предопределенными командами.

Значения Следующие стандартные команды определены в Turbo Vision и используются стандартными объектами Turbo Vision:

Таблица 4.3

Коды стандартных команд

Команда	Значение	Назначение
cmValid	0	Передается в TView.Valid для проверки вновь созданных образцов видимых элементов
cmQuit	1	Заставляет TProgram.HandleEvent вызывать EndModal (cmQuit), завершая программу. Строка статуса или одно из меню обычно содержат элемент, который переводит kbAltX и cmQuit

Команда	Значение	Назначение
cmError	2	Не обрабатывается никаким объектом. Может быть использована для представления нереализованных или неподдерживаемых команд.
cmMenu	3	Заставляет TMenuView.HandleEvent вызывать ExecView для процесса выбора меню, в результате чего может быть сгенерирована новая команда с помощью PutEvent.
cmClose	4	Обрабатывается TWindow.HandleEvent, если поле InfoPtr записи события установлено в nil или указывает на окно. Если окно модальное, то посредством PutEvent генерируется evCommand со значением из cmCancel. Если окно немодальное, то то вызывается метод Close при условии что окно поддерживает закрытие (смотри флаг wfClose). Отметка на закрывающей кнопке окна генерирует событие evCommand с Command из cmClose и InfoPtr, который указывает на окно. Строка статуса или одно из меню обычно содержит элемент, который переводит kbAltF3 в cmClose.

Команда	Значение	Назначение
cmZoom	5	Заставляет TWindow.HandleEvent вызывать TWindow.Zoom, если окно поддерживает масштабирование (смотри флаг wfZoom) и если поле InfoPtr записи события установлено в nil или указывает на окно. Отметка на кнопке масштабирования окна или двойная отметка на полосе заголовка окна генерирует событие evCommand с Command из cmZoom и InfoPtr, который указывает на окно. Строка статуса или одно из меню обычно содержит элемент, который переводит kbF5 в cmZoom.
cmResize	6	Заставляет TWindow.HandleEvent вызывать TView.DragView, если окно поддерживает изменение размеров (смотри флаги wfMove и wfGrow). Строка статуса или одно из меню обычно содержит элемент, который переводит kbCtrlF5 в cmResize.
cmNext	7	Заставляет TDeskTop.HandleEvent сдвигать последнее окно на панели экрана на передний план. Строка статуса или одно из меню обычно содержит элемент, который переводит kbF6 в cmNext.
cmPrev	8	Заставляет TDeskTop.HandleEvent сдвигать первое окно на панели экрана на самый задний план. Строка статуса или одно из меню обычно содержит элемент, который переводит kbShiftF6 в cmPrev.

Следующие стандартные команды используются для определения поведения по умолчанию объектов диалогового окна:

Таблица 4.4

Стандартные команды диалогового окна

Команда	Значение	Назначение
cmOK	10	Была нажата кнопка OK
cmCancel	11	Диалоговое окно было отменено кнопкой Cancel, закрывающей кнопкой или клавишей Enter
cmYes	12	Была нажата кнопка Yes
cmNo	13	Была нажата кнопка No
cmDefault	14	Была нажата кнопка по умолчанию

События с командами cmOK, cmCancel, cmYes или cmNo завершают модальный диалог TDialog.HandleEvent и возвращают значение команды (вызывая EndModal). Модальный диалог обычно содержит по крайней мере один TButton с одним из этих значений команд. TDialog.HandleEvent будет генерировать команду-событие cmCancel в ответ на событие от клавиатуры kbEsc. Команда cmDefault заставляет TButton.HandleEvent для умалчиваемой кнопки (см. флаг bfDefault) симулировать нажатие кнопки. TDialog.HandleEvent будет генерировать событие команды cmDefault в ответ на событие клавиатуры kbEnter.

Определены следующие стандартные команды для использования стандартными видимыми элементами:

Таблица 4.5

Стандартные команды видимых элементов

Команда	Значение	Назначение
cmReceivedFocus	50	TView.SetState использует функцию Message
cmReleasedFocus	51	Для передачи события evBroadcast с одним из этих значений в свой TView.Owner, как только sfFocused изменяется. InfoPtr события указывает на сам видимый элемент. Это информирует любой равный видимый элемент, что видимый элемент получил или освободил активность и что они должны корректировать себя соответственно. Объект TLabel, например, реагирует на эти команды, включая или выключая свою подсветку.
cmCommandSetChanged	52	Метод TProgram.Idle генерирует событие evBroadcast как только он обнаружит изменение в текущем наборе команд (вызывая методы EnableCommands, DesableCommands или SetCommands для TView). Общее сообщение cmCommandSetChanged посылается в HandleEvent каждого видимого элемента иерархии (если только их TView.EventMask специфически не маскируют события evBroadcast).

Команда	Значение	Назначение
		Если изменения в наборе команд затрагивают появление видимого элемента, он должен реагировать на cmCommandSetChanged своей перерисовкой. Объекты TButton, TMenuView и TStatusLine, например, реагируют на эту команду, перерисовывая себя.
cmScrollBarChanged	53	TScrollBar использует функцию Message для передачи события evBroadcast с одним из этих значений в свой TView.Owner, как только мышка отмечает на полосе скроллинга. InfoPtr события указывает на полосу скроллинга. Общие сообщения создаются любыми равными видимыми элементами, управляемыми полосой скроллинга, такими как объекты TScroller и TListViewer.
cmScrollBar-Clicked	54	
cmSelectWindowNum	55	Заставляет TWiondow.HandleEvent вызывать TView.Select, если InfoInt записи события соответствует TWindow.Number. TProgram.HandleEvent реагирует на события от клавиатуры от Alt-1 до Alt-9 общим сообщением cmSelectWindowNum с InfoInt от 1 до 9.

Команда	Значение	Назначение
cmRecordHistory	60	Заставляет объект THistory «записывать» текущее содержимое объекта TInputLine. TButton посылает общее сообщение cmRecordHistory своему владельцу, когда он выбран, в результате, заставляя «записывать» все объекты THistory в диалоговом окне.

См. также TView.HandleEvent, TCommandSet

Константы coXXXX

Objects

Функция Константы coXXXX передаются как параметр Code в метод TColection.Error, когда TCollection обнаруживает ошибку во время операции.

Значения Следующие стандартные коды ошибок определены для всех коллекций Turbo Vision:

Таблица 4.6

Коды ошибок коллекции

Код ошибки	Значение	Назначение
coIndexError	−1	Индекс вышел за диапазон. Параметр Info передает в метод Error содержимое неверного индекса.
coOverflow	−2	Переполнение коллекции. TCollection.SetLimit недостаточно для расширения коллекции. Параметр Info передает в метод Error запрашиваемый размер коллекции.

См. также TCollection

Объявление function CStrLen(S: String): Integer;
Функция Возвращает длину строки S, где S — это управляющая строка, использующая символы "~" для указания символов короткого набора. "~" исключаются из длины строки, поскольку они будут появляться на экране. Например, для строки '~B~roccoly', CStrLen возвращает 8.
См. также MoveCStr

Переменная CtrlBreakHit Drivers

Объявление CtrlBreakHit: Boolean = False;
Функция Драйвер обработки прерываний от клавиатуры Turbo Vision всегда устанавливает ее в True, когда нажата Ctrl-Break. Это позволяет программам на Turbo Vision перехватывать и реагировать на Ctrl-Break. Флаг может быть очищен в любое время установкой его в False.
См. также SaveCtrlBreak

Функция CtrlToArrow Drivers

Объявление function CtrlToArrow(KeyCode: Word): Word;
Функция Преобразует управляющие WordStar-совместимые коды клавиатуры в соответствующие коды клавиш курсора. Если младший байт KeyCode соответствует одному из значений управляющих клавиш в таблице 4.7, результатом будет соответствующая константа kbXXXX. В противном случае KeyCode возвращается неизменным.

Таблица 4.7

Преобразование управляющих клавиш

Клавиша	Lo(KeyCode)	Результат
Ctrl-A	$01	kbHome
Ctrl-D	$04	kbRight

Клавиша	Lo(KeyCode)	Результат
Ctrl-E	$05	kbUp
Ctrl-F	$06	kbEnd
Ctrl-G	$07	kbDel
Ctrl-S	$13	kbLeft
Ctrl-V	$16	kbIns
Ctrl-X	$18	kbDown

Переменная CursorLines — Drivers

Объявление CursorLines: Word;
Функция Устанавливает начальную и конечную строки курсора с помощью InitVideo. Формат предполагает функцию 1 прерывания BIOS $10 для установки типа курсора.
См. также InitVideo, TView.ShowCursor, TView.HideCursor, TView.BlockCursor, Tview.NormalCursor

Переменная DeskTop — App

Объявление DeskTop: PDeskTop = nil;
Функция Сохраняет указатель на TDeskTop программы. Переменная DeskTop инициализируется TProgram.InitDeskTop, которая вызывается TProgram.Init. Окна и диалоговые окна обычно вставляются (TGroup.Insert) или выполняются (TGroup.ExecView) на DeskTop.

Процедура DisposeMenu — Menus

Объявление procedure DisposeMenu(Menu: PMenu);
Функция Освобождает все элементы указанных меню (и все их подменю).
См. также Тип TMenu

Процедура DisposeStr — Objects

Объявление procedure DisposeStr(P:String);

Функция Освобождает строки, распределенные в куче с помощью функции NewStr.

См. также NewStr

Константы dmXXXX Views

Значения Биты DragMode определены так:

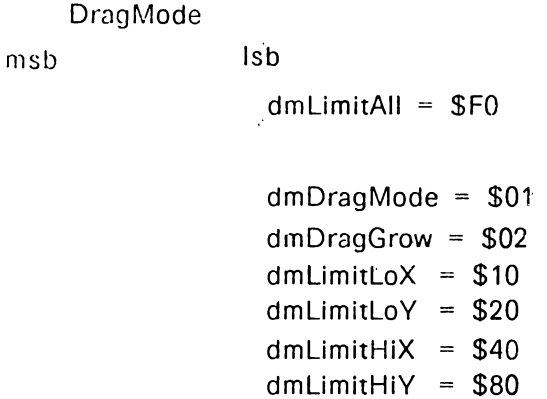


Рис. 4.1. Флаги режима Drag.

Функция Эти константы используются для компоновки параметра Mode метода TView.DragView. Они указывают разрешены ли движение и/или изменение размера и как интерпретировать параметр Limits. Константы определены так:

Таблица 4.8

Константы режима Drag

Константа	Назначение
dmDragMove	Позволяет видимому элементу перемещаться.
dmDragGrow	Позволяет видимому элементу изменять размер.

Константа	Назначение
dmLimitLoX	Левая сторона видимого элемента не может выходить за Limits.
dmLimitLoY	Верхняя сторона видимого элемента не может выходить за Limits.
dmLimitHiX	Правая сторона видимого элемента не может выходить за Limits.
dmLimitHiY	Нижняя сторона видимого элемента не может выходить за Limits.
dmLimitAll	Никакая часть видимого элемента не может выходить за Limits.

Процедура DoneEvents — Drivers

Объявление procedure DoneEvents;
Функция Завершает монитор событий Turbo Vision, отключая обработчик прерываний мышки. Вызывается автоматически при вызове TApplication.Done.
См. также TApplication.Done, InitEvents

Процедура DoneHistory — Drivers

Объявление procedure DoneHistory;
Функция Освобождает блок истории, распределенный InitHistory. Вызывается автоматически при вызове TApplication.Done.
См. также Процедура InitHistory, TApplication.Done

Процедура DoneMemory — Memory

Объявление procedure DoneMemory;
Функция Завершает монитор памяти Turbo Vision, освобождая все буфера, распределенные через GetBufMem. Вызывается автоматически при вызове TApplication.Done.
См. также TApplication.Done, InitMemory

Процедура DoneSysError — Drivers

Объявление: procedure DoneSysError;

Функция: Завершает обработчик системных ошибок Turbo Vision, восстанавливая вектора прерываний 09H, 1BH, 21H, 23H, 24H и восстанавливая состояние Ctrl-Break в DOS. Вызывается автоматически при вызове TApplication.Done.

См. также: TApplication.Done, InitSysError

Процедура DoneVideo — Drivers

Объявление: procedure DoneVideo;

Функция: Завершает монитор экрана Turbo Vision, восстанавливая начальный режим экрана (StartupMode), очищая экран и восстанавливая курсор. Вызывается автоматически при вызове TApplication.Done.

См. также: TApplication.Done, InitVideo, переменная StartupMode

Переменная DoubleDelay — Drivers

Объявление: DoubleDelay: Word = 8;

Функция: Определяет временной интервал (в 1/18.2 секундах) между нажатиями кнопки мышки для установления различия между двойным нажатием и двумя отдельными нажатиями. Используется GetMouseEvent для генерации события Double, если нажатия произошли в этом временном интервале.

См. также: TEvent.Double, GetMouseEvent

Переменная EmsCurHandle — Objects

Объявление: EmsCurhandle: Word = $FFFF;

Функция: Содержит текущий обработчик EMS, отображенный TEmsStream в нулевую физическую страницу EMS. TEmsStream избегает дорогих вызовов переотображения EMS подкачкой состояния EMS. Если ваша программа использу-

ет EMS для других целей, установите EmsCurHandle и EmsCurPage в $FFFF перед использованием TEmsStream — это будет вынуждать TEmsStream восстанавливать свое отображение.

См. также TEmsStream.Handle

Переменная EmsCurPage — Objects

Объявление EmsCurpage: Word = $FFFF;

Функция Содержит текущий номер логической страницы EMS, отображенной TEmsStream в нулевую физическую страницу EMS. TEmsStream избегает дорогих вызовов переотображения EMS подкачкой состояния EMS. Если ваша программа использует EMS для других целей, установите EmsCurHandle и EmsCurPage в $FFFF перед использованием TEmsStream — это будет вынуждать TEmsStream восстанавливать свое отображение.

См. также TEmsStream.Page

Константы evXXXX — Drivers

Функция Эти мнемоники обозначают типы событий для обработчиков событий Turbo Vision. Константы evXXXX используются в нескольких местах: в поле What записи события, в поле EventMask видимого элемента и в переменных PositionalEvents и FocusedEvents.

Значения Следующие значения флагов событий обозначают стандартные типы событий:

Таблица 4.9

Флаги стандартных событий

Константа	Значение	Назначение
evMouseDown	$0001	Кнопка мышки нажата
evMouseUp	$0002	Кнопка мышки отпущена

Константа	Значение	Назначение
evMouseMove	$0004	Мышка изменила положение
evMouseAuto	$0008	Периодическое событие до тех пор, пока нажата кнопка мышки
evKeyDown	$0010	Клавиша нажата
evCommand	$0100	Событие-команда
evBroadcast	$0200	Событие-общее сообщение

Следующие константы могут использоваться для маскирования типов событий:

Таблица 4.10

Маски стандартных событий

Константа	Значение	Назначение
evNothing	$0000	Событие уже обработано
evMouse	$000F	Событие от мышки
evKeyboard	$0010	Событие от клавиатуры
evMessage	$FF00	Событие-сообщение (команда, общее сообщение или определено пользователем).

Биты маски события определены на рис. 4.2. Маски стандартных событий могут быть использованы для быстрого определения, принадлежит ли событие конкретному семейству событий. Например,

```
if Event.What and evMouse <> 0
   then DoMouseEvent;
```

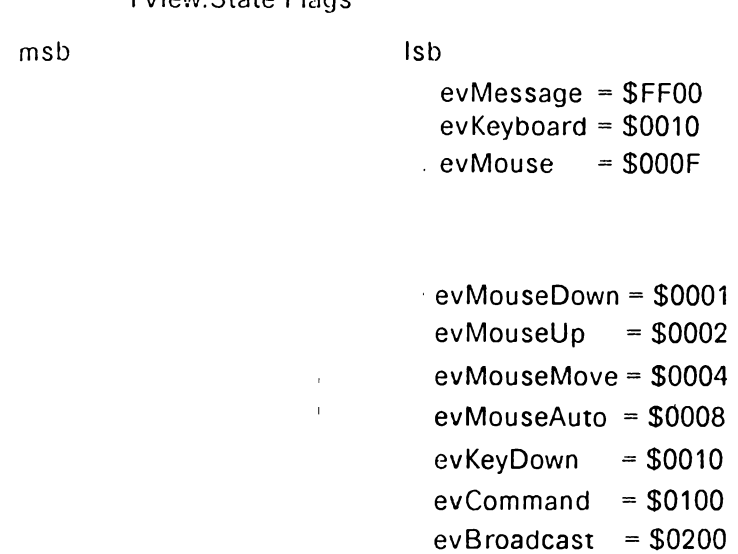


Рис. 4.2. Отображение битов маски события.

См. также TEvent, TView.EventMask, GetKeyEvent, GetMouseEvent, методы HandleEvent, PositionalEvents, FocusedEvents.

Тип FNameStr ???

Объявление FNameStr = String[79];
Функция Строка, содержащая имя файла DOS.

Переменная FocusedEvents Views

Объявление FocusedEvents: Word = evKeyboard + evCommand;
Функция Определяет классы событий как сфокусированные события. Переменные FocusedEvents и PositionalEvents используются TGroup.HandleEvent для определения, как соотносятся события к подэлементам группы.

Если класс события не содержится в FocusedEvents или PositionalEvents, то оно интерпретируется как общее событие.

См. также Переменные PositionalEvents, TGroup.HandleEvent, TEvent, константы evXXXX.

Процедура FormatStr — Drivers

Объявление `procedure FormatStr(var Result: String; Format: String; var Params);`

Функция Процедура форматирования строки, которая работает подобно функции языка Си vsprintf. Format включает спецификаторы формата, а Params содержит список параметров. FormatStr выполняет форматированный вывод строки в Result.

Параметр Format может содержать любое число спецификаторов формата, для отображения параметров в Params. Формат спецификаторов — % |- ||nnn |X, где

- % указывает начало спецификатора формата;
- |-| необязательный знак минуса, указывающий, что параметр будет выровнен влево (по умолчанию параметры при отображении выравниваются вправо);
- |nnn| — необязательный десятичный спецификатор длины в диапазоне 0–255 (0 указывает на отсутствие длины, а не нуль означает, что выводится поле в nnn символов);
- X — символ формата:
- 's' означает, что параметр указывает на строку;
- 'd' означает десятичное представление LongInt параметра;
- 'c' означает, что младший байт параметра — символ;
- 'x' означает шестнадцатиричное представление параметра LongInt.
- '#'устанавливает индекс параметра в nnn.

Например, если параметр указывает на строку, содержащую 'spiny', следующая таблица

показывает спецификаторы и их результаты при печати:

Таблица 4.11

Спецификаторы формата и их результаты

Спецификатор	Результат
%6s	' spiny'
%-6s	'spiny'
%3s	'iny'
%-3s	'spi'
%06s	'0spiny'
%-06s	'spiny0'

Params — это нетипизированный var параметр, содержащий параметры с соответствующими спецификаторами формата в Format. Params должен быть массивом из LongInt или указателей или записью, содержащей LongInt или указатели. Например, для вывода строки сообщения об ошибке

```
Error in file [file name] at line [line number]
```

Вы должны послать следующую строку в Format:

```
'Error in file %s at line %d'.
```

Params должен содержать указатель на строку имени файла и Longint, представляющая число строк в файле. Это может быть сделано двумя способами: в массиве или в записи. Следующий пример показывает два типа объявлений и присвоений переменных, оба создают допустимые значения, передаваемые как Params в FormatStr.

```
type
  ErrMsgRec = record
```

```
FileName: PString;
LineNo: Longint;
end;

ErrMsgArray = array[0..1] of Longint;

const
TemplateMsg = 'Error in file %s at line %d';

var
MyFileName: FNameStr;
OopsRec: ErrMsgRec;
DarnArray: ErrMsgArray;
TestStr: String;

begin
MyFileName := 'WARTHOG.ASM';

with OopsRec do
begin
FileName := @MyFileName;
LineTo := 42;
end;
FormatStr(TestStr, TemplateMsg, OopsRec);
Writeln(TestStr);

DarnArray[0] := Longint(@MyFileName);
DarnArray[1] := 24;
FormatStr(TestStr, TemplateMsg, DarnArray);
Writeln(TestStr);
end;
```

См. также Функцию SystemError, объект TParamText.

Процедура FreeBufMem — Memory

Объявление `procedure FreeBufMem(P: Pointer);`

Функция Освобождает кэш-буфер, ссылаемый указателем P.

См. также GetBufMem, DoneMemory.

Функция GetAltChar Drivers

Объявление function GetAltChar(KeyCode: Word): Char;
Функция Возвращает символ Ch, для которого Alt-Ch вырабатывает двухбайтовый скэн-код, данный в аргументе KeyCode. Эта функция дает обратное к GetAltCode отображение.
См. также GetAltCode.

Функция GetAltCode Drivers

Объявление function GetAltCode(Ch: Char): Word;
Функция Возвращает двухбайтовый скэн-код, соответствующий Alt-Ch. Эта функция делает обратное к GetAltChar отображение.
См. также GetAltChar.

Процедура GetBufMem Memory

Объявление procedure GetBufMem(var P: Pointer; Size: Word);
Функция Распределяет кэш-буфер для Size байт и запоминает указатель на него в P. Если нет памяти для кэш-буфера запрашиваемого размера, P устанавливается в nil. Кэш- буфер отличается от обычных блоков кучи (распределяемых с помощью New, GetMem или MemAlloc), в которых они могут размещаться или освобождаться монитором памяти в любое время. Указатель, передаваемый в GetBufMem, становится указателем на кэш-буфер и он (и только он) корректируется, когда буфер перемещается монитором памяти. Если монитор памяти решает освободить буфер, он устанавливает этот указатель в nil. Кэш-буфер может быть освобожден через вызов FreeBufMem. Кэш-буфера будут занимать любое нераспределенное пространство кучи между HeapPtr и HeapEnd, включая область, установленную для пула надежности программы.

Turbo Vision использует кэш-буфера для подкачки содержимого объектов TGroup (таких, как окна, диалоговые окна и панель экрана), как только эти объекты устанавливают флаг ofBuffered — это значительно повышает производительность операций перерисовки.

См. также FreeBuffMem, InitMemory, TGroup.Draw.

Процедура GetKeyEvent — Drivers

Объявление procedure GetKeyEvent(var Event: TEvent);

Функция Проверяет, доступно ли событие от клавиатуры вызовом прерывания BIOS INT 16H. Если клавиша была нажата, Event.What устанавливается в evKeyDown и Event.KeyCode устанавливается в скэн-код клавиши. В противном случае, Event.What устанавливается в evNothing. GetKeyEvent вызывается из TProgram.GetEvent.

См. также TProgramm.GetEvent, константы evXXXX, TView.HandleEvent.

Процедура GetMouseEvent — Drivers

Объявление procedure GetMouseEvent(var Event: TEvent);

Функция Проверяет, доступно ли событие от мышки из очереди событий от мышки, поддерживаемой обработчиком событий Turbo Vision. Если происходит событие от мышки, Event.What устанавливается в evMouseDown, evMouseUp, evMouseMove или evMouseAuto; Event.Buttons устанавливается в mbLeftButton или mbRightButton; Event.Double устанавливается в True или False; Event.Where устанавливается в позицию мышки в глобальных координатах (соответствующих координатной системе TApplication). Если события от мышки недоступны, Event.What устанавливается в evNothing. GetMouseEvent вызывается из TProgram.GetEvent.

См. также TProgram.GetEvent, события evXXXX, методы HandleEvent.

Константы gfXXXX Views

Функция Эти мнемоники используются для установки полей GrowMode во всех объектах TView и порожденных. Биты, установленные в GrowMode, определяют, как видимый элемент будет изменяться в зависимости от изменений размера его владельца.

Значения Биты GrowMode определены как:

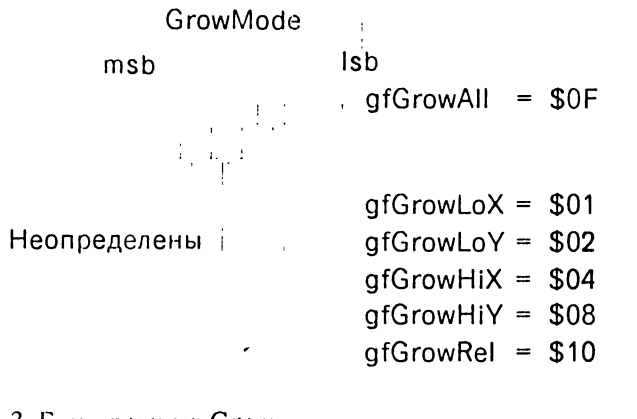


Рис. 4.3. Биты режима Grow

Таблица 4.12

Определения флага режима **Grow**

Константа	Назначение
gfGrowLoX	Если установлен, то левая сторона видимого элемента будет находиться на одном расстоянии от правой стороны владельца.

Константа	Назначение
gfGrowLoY	Если установлен, то верхняя сторона видимого элемента будет находиться на одном расстоянии от нижней стороны владельца.
gfGrowHiX	Если установлен, то правая сторона видимого элемента будет находиться на одном расстоянии от правой стороны владельца.
gfGrowHiY	Если установлен, то нижняя сторона видимого элемента будет находиться на одном расстоянии от нижней стороны владельца.
gfGrowAll	Если установлен, то видимый элемент будет сдвигаться вместе с правым нижним углом его владельца.
gfGrowRel	Для использования с объектами TWindow, которые находятся в панели экрана: видимый элемент будет изменять размер относительно размера владельца. Окно будет обрабатываться соответственно размеру владельца, даже когда происходит переключение между режимами 25 и 43/50 строк.

См. также TView.GrowMode

Константы hcXXXX Views

Значения Определены следующие константы контекста подсказки:

Таблица 4.13

Константы контекста подсказки

Константа	Значение	Назначение
hcNoContext	0	Контекст не задан
hcDragging	1	Объект — перемещаем

Функция Значение TView.HelpCtx по умолчанию — hcNoContext, которое указывает, что для видимого элемента нет контекста подсказки. TView.GetHelpCtx возвращает hcDragging, как только видимый элемент становится перемещаемым (это указывается состоянием флага sfDragging).
Turbo Vision резервирует для контекста подсказки значения от 0 до 999. Программисты могут определять свои константы в диапазоне от 1000 до 65535.

См. также TView.HelpCtx, TStatusLine.Update.

Процедура HideMouse Drivers

Объявление procedure HideMouse;

Функция Курсор мышки изначально видим после вызова InitEvents. HideMouse прячет мышку и увеличивает внутренний «счетчик мышки» в драйвере мышки. ShowMouse будет уменьшать этот счетчик и показывать курсор мышки, когда счетчик становится равен 0. Таким образом, вызовы HideMouse и ShowMouse могут быть вложенными, но всегда должны быть сбалансированы.

См. также InitEvents, DoneEvents, ShowMouse

Переменная HiResScreen Drivers

Объявление HiResScreen: Boolean;

Функция Устанавливается в True с помощью InitVideo, если экран поддерживает режим 43/50 строк (EGA/VGA); в противном случае устанавливается в False.

См. также InitVideo

Процедура HistoryAdd HistList

Объявление procedure HistoryAdd(Id: Byte; var Str: String);

Функция Добавляет строку Str в список истории, указываемый с помощью Id.

Переменная HistoryBlock — HistList

Объявление HistoryBlock: Pointer = nil;

Функция Указывает на буфер, вызывающий блок истории и используемый для хранения строк истории. Размер блока определяется посредством HistorySize. Указатель устанавливается в nil до тех пор пока не будет установлен с помощью InitHistory. Его значение нельзя изменить.

См. также процедуру InitHistory, переменную HistorySize.

Функция HistoryCount — HistList

Объявление function HistoryCount(Id: Byte): Word;

Функция Возвращает количество строк в списке истории, соответствующее номеру ID.

Переменная HistorySize — HistList

Объявление HistorySize: Word = 1024;

Функция Задает размер блока истории, используемый администратором списка истории для хранения значений, введенных в строках ввода. Размер фиксируется посредством InitHistory при запуске программы. По умолчанию размер блока равен 1K, но может быть изменен перед вызовом InitHistory. Значение нельзя изменять после вызова InitHistory.

См. также процедуру InitHistory, переменную HistoryBlock.

Функция HistoryStr — HistList

Объявление function HistoryStr(Id: Byte; Index: Integer): String;

Функция Возвращает Index строку в списке истории, соответствующую номеру ID.

Переменная HistoryUsed — HistList

Объявление HistoryUsed: Word = 0;

Функция Используется внутри администратором списка истории для указания на смещение внутри блока истории. Это значение нельзя изменить.

Процедура InitEvents — Drivers

Объявление procedure InitEvents;
Функция Инициализирует монитор событий Turbo Vision, подключая обработчик прерываний мышки и показывая мышку. Вызывается автоматически TApplication.Init.
См. также DoneEvents.

Процедура InitHistory — HistList

Объявление InitHistory;
Функция Вызывается с помощью TApplication.Init для распределения блока памяти в куче, используемом монитором списка истории. Размер блока определяется переменной HistorySize. После вызова InitHistory переменная HistoryBlock указывает на начало блока.
См. также TProgram.Init, процедуру DoneHistory.

Процедура InitMemory — Memory

Объявление procedure InitMemory;
Функция Инициализирует монитор памяти Turbo Vision, инсталлируя функцию объявления кучи в HeapError. Вызывается автоматически посредством TApplication.Init.
См. также DoneMemory.

Процедура InitSysError — Drivers

Объявление procedure InitSysError;
Функция Инициализирует обработчик системных ошибок Turbo Vision, переопределяя вектора прерываний 09H, 1BH,21H, 23H, 24H и очищая состояние Ctrl-Break в DOS. Вызывается автоматически посредством TApplication.Init.

См. также DoneSysError.

Процедура InitVideo — Drivers

Объявление procedure InitVideo;

Функция Инициализирует монитор экрана Turbo Vision. Сохраняет текущий режим экрана в StartupMode и переключает экран в режим, указанный в ScreenMode. Переменные ScreenWidth, ScreenHeight, HiResScreen, CheckSnow, ScreenBuffer и CursorLines корректируются соответственно. Режим экрана позднее может быть изменен использованием SetVideoMode. InitVideo вызывается автоматически посредством TApplication.Init.

См. также DoneVideo, SetVideoMode, smXXXX.

Константы kbXXXX — Drivers

Функция Два множества констант, начинающихся с "kb", связаны с клавиатурой.

Значения Следующие значения определяют состояние клавиатуры и могут быть использованы при проверке регистров Shift клавиатуры, которое запоминается в байте с абсолютным адресом $40:$17. Например,

```
var
ShiftState: Byte absolute $40:$17;
...
if ShiftState and kbAltShift <> 0
 then AltKeyDown;
```

Таблица 4.14

Состояние клавиатуры и маски Shift

Константа	Значение	Назначение
kbRightShift	$0001	Установлено, если правый Shift нажат

Константа	Значение	Назначение
kbLeftShift	$0002	Установлено, если левый Shift нажат
kbCtrlShift	$0004	Установлено, если Ctrl нажат
kbAltShift	$0008	Установлено, если Alt нажат
kbScrollState	$0010	Установлено, если клавиатура в состоянии ScrollLock
kbNumState	$0020	Установлено, если клавиатура в состоянии NumLock
kbCapsState	$0040	Установлено, если клавиатура в состоянии CapsLock
kbInsState	$0080	Установлено, если клавиатура в состоянии InsLock

Таблица 4.15

Коды Alt-буква

Константа	Значение	Константа	Значение
kbAltA	$1E00	kbAltN	$3100
kbAltB	$3000	kbAltO	$1800
kbAltC	$2E00	kbAltP	$1900
kbAltD	$2000	kbAltQ	$1000
kbAltE	$1200	kbAltR	$1300
kbAltF	$2100	kbAltS	$1F00
kbAltG	$2200	kbAltT	$1400
kbAltH	$2300	kbAltU	$1600
kbAltI	$1700	kbAltV	$2F00
kbAltJ	$2400	kbAltW	$1100
kbAltK	$2500	kbAltX	$2D00
kbAltL	$2600	kbAltY	$1500

Константа	Значение	Константа	Значение
kbAltM	$3200	kbAltZ	$2C00

Таблица 4.16

Коды специальных клавиш

Константа	Значение	Константа	Значение
kbAltEqual	$8300	kbEnd	$4F00
kbAltMinus	$8200	kbEnter	$1C0D
kbAltSpace	$0200	kbEsc	$011B
kbBack	$0E08	kbGrayMinus	$4A2D
kbCtrlBack	$0E7F	kbHome	$4700
kbCtrlDel	$0600	kbIns	$5200
kbCtrlEnd	$7500	kbLeft	$4B00
kbCtrlEnter	$1C0A	kbNoKey	$0000
kbCtrlHome	$7700	kbPgDn	$5100
kbCtrlIns	$0400	kbPgUp	$4900
kbCtrlLeft	$7300	kbrayPlus	$4E2B
kbCtrlPgDn	$7600	kbRight	$4D00
kbCtrlPgUp	$8400	kbShiftDel	$0700
kbCtrlPrtSc	$7200	kbShiftIns	$0500
kbCtrlRight	$7400	kbShiftTab	$0F00
kbDel	$5300	kbTab	$0F09
kbDown	$5000	kbUp	$4800

Таблица 4.17

Коды Alt-число

Константа	Значение	Константа	Значение
kbAlt1	$7800	kbAlt6	$7D00
kbAlt2	$7900	kbAlt7	$7E00

Константа	Значение	Константа	Значение
kbAlt3	$7A00	kbAlt8	$7F00
kbAlt4	$7B00	kbAlt9	$8000
kbAlt5	$7C00	kbAlt0	$8100

Таблица 4.18

Коды функциональных клавиш

Константа	Значение	Константа	Значение
kbF1	$3B00	kbF6	$4000
kbF2	$3C00	kbF7	$4100
kbF3	$3D00	kbF8	$4200
kbF4	$3E00	kbF9	$4300
kbF5	$3F00	kbF0	$4400

Таблица 4.19

Коды Shift-функциональная клавиша

Константа	Значение	Константа	Значение
kbShiftF1	$5400	kbShiftF6	$5900
kbShiftF2	$5500	kbShiftF7	$5A00
kbShiftF3	$5600	kbShiftF8	$5B00
kbShiftF4	$5700	kbShiftF9	$5C00
kbShiftF5	$5800	kbShiftF0	$5D00

Таблица 4.20

Коды Ctrl-функциональная клавиша

Константа	Значение	Константа	Значение
kbCtrlF1	$5E00	kbCtrlF6	$6300
kbCtrlF2	$5F00	kbCtrlF7	$6400
kbCtrlF3	$6000	kbCtrlF8	$6500

Константа	Значение	Константа	Значение
kbCtrlF4	$6100	kbCtrlF9	$6600
kbCtrlF5	$6200	kbCtrlF0	$6700

Таблица 4.21.

Коды Alt-функциональная клавиша

Константа	Значение	Константа	Значение
kbAltF1	$6800	kbAltF6	$6D00
kbAltF2	$6900	kbAltF7	$6E00
kbAltF3	$6A00	kbAltF8	$6F00
kbAltF4	$6B00	kbAltF9	$7000
kbAltF5	$6C00	kbAltF0	$7100

См. также evKeyDown, GetKeyEvent.

Функция LongDiv Objects

Объявление function LongDiv(X: Longint; Y: Integer): Integer; inline($59/$58/$5A/$F7/$F9);

Функция Функция со встроенным ассемблерным кодом для быстрого деления, возвращающая целое значение X/Y.

Функция LongMul Objects

Объявление function LongMul(X, Y: Integer): Longint; inline($5A/$58/$F7/$EA);

Функция Функция со встроенным ассемблерным кодом для быстрого умножения, возвращающая длинное целое значение X*Y.

Тип LongRec Objects

Объявление LongRec = record Lo, Hi: Word; end;

Функция Полезный тип записи для обработки переменных двойного слова.

Функция LowMemory Memory

Объявление function LowMemory: Boolean;

Функция Возвращает True, если памяти мало, в противном случае False. True означает, что вызов распределения памяти достиг пула надежности. Размер пула надежности определяется переменной LowMemSize.

См. также Главу 6, InitMemory, TView.Valid, LowMemSize.

Переменная MaxBufMem — Memory

Объявление MaxBufMem: Word = 65536 div 16;

Функция Указывает максимальный объем памяти в 16-байтовых параграфах, которая может быть распределена для кэш-буферов.

См. также GetBufMem, FreeBufMem.

Переменная MaxCollectionSize — Objects

Объявление MaxCollectionSize = 65520 div SizeOf(Pointer);

Функция Определяет максимальное число элементов, которые может содержать коллекция, по существу это число указателей, которое помещается в сегмент памяти в 64K.

Константа MaxViewWidth — Views

Объявление MaxViewWidth = 132;

Функция Устанавливает максимальную длину видимого элемента.

См. также поле TView.Size.

Константа mbXXXX — Drivers

Функция Эти константы могут быть использованы при проверке поля TEvent.Buttons записи события evMouse.

```
if (Event.What = evMouseDown) and
   (Event.Button = mbLeftButton)
then LeftButtonDown;
```

Значения Определены следующие константы:

Таблица 4.22

Константы кнопки мышки.

Константа	Значение	Назначение
mbLeftButton	$01	Установлено, если была нажата левая кнопка
mbRightButton	$02	Установлено, если была нажата правая кнопка

См. также GetMouseEvent.

Функция MemAlloc — Memory

Объявление function MemAlloc(Size: Word): Pointer;

Функция Распределяет Size байт памяти в куче и возвращает указатель на блок. Если блок требуемого размера не может быть распределен, возвращается значение nil. В отличие от стандартных процедур New и GetMem, MemAlloc не позволяет распределять пул надежности. Блок распределенный с помощью MemAlloc может быть освобожден стандартной процедурой FreeMem.

См. также New, GetMem, Dispose, FreeMem, MemAllocSeg.

Функция MemAllocSeg — Memory

Объявление function MemAllocSeg(Size: Word): Pointer;

Функция Распределяет блок памяти, выровненный на границу сегмента. Соответствует MemAlloc, за исключением того, что часть смещения результирующего значения указателя равна 0.

См. также MemAlloc

Переменная MenuBar — App

Объявление MenuBar: PMenuView = nil;

Функция Сохраняет указатель на полосу меню программы (наследник TMenuView). Переменная MenuBar инициализируется с помощью

TProgram.InitMenuBar, которая вызывается через TProgram.Init. Значение nil указывает, что программа не имеет полосы меню.

Функция Message — Views

Объявление: function Message(Receiver: PView; What, Command: Word; InfoPtr: Pointer): Pointer;

Функция: Устанавливает событие-команду с аргументами What, Command или InfoPtr, затем, если возможно, вызывает Receiver^.HandleEvent для обработки этого события. Message возвращает nil, если Receiver — nil или, если событие не обработано успешно. Если событие успешно обработано (HandleEvent возвращает Event.What как evNothing), функция возвращает Event.InfoPtr. Последний может быть использован для определения, каким видимым элементом обработано событие. Аргумент What обычно устанавливается в evBroadcast. Например, по умолчанию TscrollBar.ScrollDraw посылает следующее сообщение в полосу скроллинга владельца:

```
Message(Owner, evBroadcast,
                cmScrollBarChanged, @Self);
```

Это сообщение гарантирует, что соответствующие видимые элементы перерисуются как только значение полосы скроллинга изменится.

См. также: TView.HandleEvent, тип TEvent, константы cmXXXX, константы evXXXX.

Переменная MinWinSize — Views

Объявление: MinWinSize: TPoint = (X: 16; Y: 6);

Функция: Определяет минимальный размер TWindow или его потомков. Значение возвращается в параметре при вызове TWindow.SizeLimits. Любые изменения в MinWinSize будут воздействовать на все окна, если только метод SizeLimits окна не перекрыт.

См. также: TWindow.SizeLimits

Переменная MouseButtons Drivers

Объявление MouseButtons: Byte;

Функция Содержит текущее состояние кнопок мышки. MouseButtons корректируется обработчиком прерываний мышки как только кнопка нажата или отпущена. Константы mbXXXX могут быть использованы для проверки MouseButtons.

См. также константы mbXXX.

Переменная MouseEvents Drivers

Объявление MouseEvents: Boolean = False;

Функция Устанавливается в True, если мышка инсталлирована и обнаружена InitEvents; в противном случае, устанавливается в False. Если False, то все процедуры событий от мышки обходятся.

См. также GetMouseEvent.

Переменная MouseIntFlag Drivers

Объявление MouseIntFlag: Byte;

Функция Используется внутри драйвера мышки Turbo Vision и видимыми элементами. Устанавливается как только возникает событие от мышки.

Переменная MouseWhere Drivers

Объявление MouseWhere: TPoint;

Функция Содержит текущую позицию мышки в глобальных координатах. MouseWhere корректируется обработчиком прерываний мышки как только мышка сдвигается. Используйте процедуру MakeLocal для преобразования к локальным (относительно окна) координатам. MouseWhere передается в обработчики событий вместе с другими данными мышки.

См. также GetMouseEvent, методы GetEvent, MakeLocal

Процедура MoveBuf — Objects

Объявление: procedure MoveBuf(var Dest; var Source; Attr: Byte; Count: Word);

Функция: Копирует текст в буфер для использования с TView.WriteBuf или TView.WriteLine. Dest должен быть TDrawBuffer (или эквивалентным массивом слов) и Source должен быть массивом байт. Count байт копируются из Source в младшие байты соответствующих слов в Dest. Старшие байты слов в Dest устанавливаются в Attr или остаются неизменными, если Attr равен 0.

См. также: тип TDrawBuffer, MoveChar, MoveCStr, MoveStr.

Процедура MoveChar — Objects

Объявление: procedure MoveChar(var Dest; C: Char; Attr: Byte; Count: Word);

Функция: Копирует символы в буфер для использования с TView.WriteBuf или TView.WriteLine. Dest должен быть TDrawBuffer (или эквивалентным массивом слов). Младшие байты первых Count слов Dest устанавливаются в C или остаются неизменными, если Ord(C) — 0. Старшие байты слов устанавливаются в Attr или остаются неизменными, если Attr — 0.

См. также: тип TDrawBuffer, MoveBuf, MoveCStr, MoveStr.

Процедура MoveCStr — Objects

Объявление: procedure MoveCStr(var Dest; Str: String; Attrs: Word);

Функция: Копирует строку в буфер для использования с TView.WriteBuf или TView.WriteLine. Dest должен быть TDrawBuffer (или эквивалентным массивом слов). Символы в Str копируются в младшие байты соответствующих слов в Dest. Старшие байты слов устанавливаются

в Lo(Attr) или в Hi(Attr). Символы "~" в строке используются для переключения между двумя байтами атрибута, передаваемыми в слове Attr.

См. также тип TDrawBuffer, MoveChar, MoveBuf, MoveStr.

Процедура MoveStr — Objects

Объявление procedure MoveStr(var Dest; Str: String; Attr: Byte);

Функция Копирует строку в буфер для использования с TView.WriteBuf или TView.WriteLine. Dest должен быть TDrawBuffer (или эквивалентным массивом слов). Символы в Str копируются в младшие байты соответствующих слов в Dest. Старшие байты слов устанавливаются в Attr или остаются неизменными, если Attr равен 0.

См. также тип TDrawBuffer, MoveChar, MoveCStr, MoveBuf.

Функция NewItem — Menus

Объявление function NewItem(Name, Param: TMenuStr; KeyCode: Word; Command: Word; AHelpCtx: Word; Next: PMenuItem): PMenuItem;

Функция Распределяет и возвращает указатель на новую запись TMenuItem, которая представляет элемент меню (NewStr используется для распределения полей указателей строк Name и Param). Параметр Name должен быть непустой строкой и параметр Command должен быть ненулевым. Вызовы NewItem, NewLine, NewMenu и NewSubMenu могут быть вложенными для создания полного дерева меню в одном операторе Паскаля; для примеров см. главу 2 Описания.

См. также TApplication.InitMenuBar, тип TMenuView, NewLine, NewMenu, NewSubMenu.

Объявление function NewLine(Next: PMenuItem): PMenuItem;

Функция Распределяет и возвращает указатель на новую запись TMenuItem, которая представляет отдельную строку в окне меню.

См. также TApplication.InitMenuBar, тип TMenuView, NewMenu, NewSubMenu, NewItem.

Функция NewMenu Menus

Объявление function NewMenu(Items: PMenuItem): Pmenu;

Функция Распределяет и возвращает указатель на новую запись TMenu. Поля Items и Default записи устанавливаются в значение данное параметром Items.

См. также TApplication.InitMenuBar, тип TMenuView, NewLine, NewSubMenu, NewItem.

Функция NewSItem Dialogs

Объявление function NewSItem(Str: String; ANext: PSItem): PSItem;

Функция Распределяет и возвращает указатель на новую запись TSItem. Поля Value и Next записи устанавливаются в NewStr(Str) и ANext соответственно. Функция NewSItem и запись типа TSItem позволяют легко конструировать связанные списки строк; для примера см. главу 4 Описания.

Функция NewStatusDef Menus

Объявление function NewStatusDef(AMin, AMax: Word; AItems: PStatusItem; ANext: PStatusDef): PStatusDef;

Функция Распределяет и возвращает указатель на новую запись TStatusDef. Запись инициализируется с данными значениями параметров. Вызовы NewStatusDef NewStatusKey могут

быть вложенными для создания полных описаний строк статуса в одном операторе Паскаля; для примеров см. главу 2 Описания.

См. также TApplication.InitStatusLine, TStatusLine, NewStatusKey.

Функция NewStatusKey — Menus

Объявление function NewStatusKey(AText: String; AKeyCode: Word; ACommand: Word; ANext: PStatusItem): PStatusItem;

Функция Распределяет и возвращает указатель на новую запись TStatusItem. Запись инициализируется со значениями параметров (NewStr используется для распределения поля указателя Text). Если AText пусто (результатом будет nil в поле Text), элемент статуса скрывается, но будет обеспечивать, однако, отображение из данного KeyCode в Command.

См. также TApplication.InitStatusLine, TStatusLine, NewStatusDef.

Функция NewStr — Objects

Объявление function NewStr(S: String): PString;

Функция Функция динамической строки. Если S — пустая, NewStr возвращает nil; в противном случае, распределяются Length(S)+1 байт, содержащие копию S, и возвращается указатель на первый байт.

Строки, создаваемые с помощью NewStr, могут быть освобождены с помощью DisposeStr.

См. также DisposeStr.

Функция NewSubMenu — Menus

Объявление function NewSubMenu(Name: TmenuStr; AHelpCtx: Word; SubMenu: PMenu; Next: PMenuItem): PMenuItem;

Функция Распределяет и возвращает указатель на новую запись TMenuItem, которая представляет

подменю (NewStr используется для распределения поля указателя Name).

См. также TApplication.InitMenuBar, TMenuView, NewLine, NewItem.

Константы ofXXXX

Views

Функция Эти мнемоники используются для ссылок на битовые позиции поля TView.Options. Установка позиции бита в 1 указывает, что видимый элемент имеет отдельный атрибут; очистка битовой позиции означает, что атрибут отключен или запрещен. Например,

```
MyWindow.Options := ofTileable + ofSelectable;
```

Значения Определены следующие опции флагов:

Таблица 4.23.

Опции флагов

Константа	Назначение
ofSelectable	Установлен, если видимый элемент выбирает себя автоматически (см. sfSelectable), например, отметкой мышкой в видимом элементе или клавишей Tab в диалоговом окне.
ofTopSelect	Установлен, если видимый элемент помещается перед всеми другими равными видимыми элементами, когда он выбран. Когда бит ofTopSelect установлен, вызов TView.Select соответствует вызову TView.MakeFirst. Окна (TWindow и его потомки) по умолчанию имеют этот бит установленным, что заставляет их располагаться перед всеми другими окнами на панели экрана

Константа	Назначение
ofFirstClick	Если очищен, отметка мышкой, которая выбирает видимый элемент, не имеет эффекта. Если установлен, такая отметка мышкой будет работать как обычный отметка мышкой после выбора видимого элемента. Не имеет эффекта, если ofSelectable не установлен. См. также TView.HandleEvent, sfSelect, ofSelectable.
ofFramed	Установлен, если видимый элемент имеет рамку. TWindow и его потомки имеют TFrame, как свой последний подэлемент. Когда видимый элемент рисует себя, TFrame рисует рамку вокруг любого другого подэлемента, у которого установлен бит ofFrame. См. также TFrame, TWindow.
ofPreProcess	Установлен, если видимый элемент получает активные события до того, как они были посланы активному элементу. В противном случае очищен. См. также sfFocused, ofPostProcess, TGroup.Phase.
ofPostProcess	Установлен, если видимый элемент получает активное событие в случае, когда активный элемент не может их обработать. В противном случае очищается. См. также sfFocused, ofPreProcess, TGroup.Phase.

Константа	Назначение
ofBuffered	Используется только для объектов TGroup. Установлен, если кэш-буфер распределяет доступную память. Буфер группы содержит образ экрана для всей группы, таким образом увеличивая скорость перерисовки. При отсутствии буфера, TGroup.Grow вызывает методы каждого подэлемента DrawView. Если впоследствии New и GetMem не могут получить достаточно памяти, буфера группы будут освобождать память. См. также GetBufMem.
ofTileable	Установлен, если панель экрана может расположить этот видимый элемент заполнением (или каскадом). Обычно используется только с объектами TWindow.
ofCenterX	Установлен, если видимый элемент центрируется по оси X своего владельца при вставке в группу с использованием TGroup.Insert.
ofCenterY	Установлен, если видимый элемент центрируется по оси Y своего владельца при вставке в группу с использованием TGroup.Insert.
ofCentered	Установлен, если видимый элемент центрируется по обоим осям своего владельца при вставке в группу с использованием TGroup.Insert.

Биты Options определены на рис. 4.4.

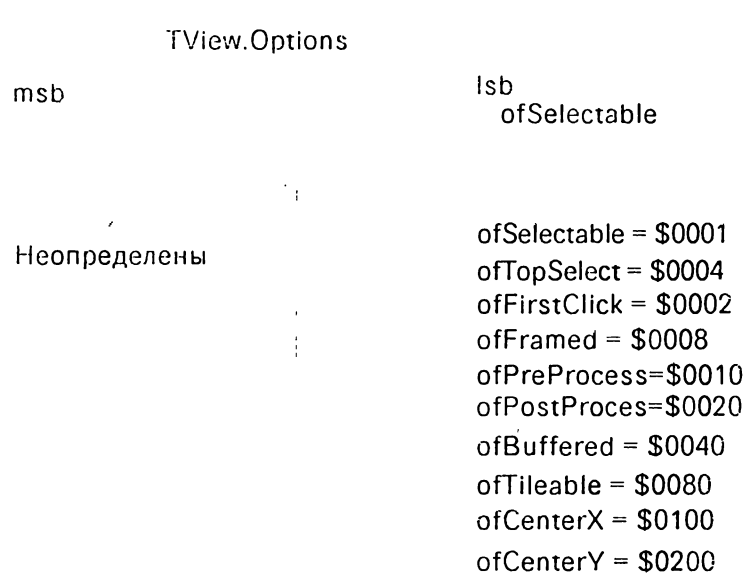


Рис. 4.4. Флаги битов Options

См. также TView.Options.

Тип PChar Objects

Объявление PChar = ^Char;

Функция Определяет указатель на символ.

Переменная PositionalEvents Views

Объявление PositionalEvents:Word = evMouse;

Функция Определяет классы событий как позиционированные события. Переменные FocusedEvents и PositionalEvents используются TGroup.HandleEvent, чтобы установить соответствие события подэлементам группы. Если класс события не принадлежит FocusedEvents или PositionalEvents, то оно интерпретируется как общее событие.

См. также TGroup.HandleEvent, тип TEvent, константы события evXXXX, переменная Focused Events.

Процедура PrintStr — Drivers

Объявление: procedure PrintStr(S: String);

Функция: Печатает строку S на экране, используя вызов функции DOS 40H для записи в стандартное устройство вывода DOS. Имеет тот же эффект, что и Write (S), за исключением того, что PrintStr не требует редактирования с программой библиотеки времени выполнения файлового ввода-вывода.

Тип PString — Objects

Объявление: PString = ^String;

Функция: Определяет указатель на строку.

Тип PtrRec — Objects

Объявление: PtrRec = record Ofs, Seg: Word; end;

Функция: Запись, содержащая значение сегмента и смещения указателя.

Процедура RegisterDialogs — Dialogs

Объявление: procedure RegisterDialogs;

Функция: Вызывает RegisterType для каждого стандартного типа объекта, определенного в модулях TDialog, TInputLine, TButton, TCluster, TRadioButtons, TCheckBoxes, TListBox, TStaticText, TParamText, TLabel, THistory. Это позволяет использовать все эти объекты с потоком ввода-вывода.

См. также: TStreamRec, RegisterTypes.

Процедура RegisterType — Objects

Объявление: procedure RegisterType(var S: TStreamRec);

Функция: Тип объекта Turbo Vision должен быть зарегистрирован перед использованием в потоке ввода-вывода. Стандартные типы объектов уже зарегистрированы с ObjTypes в резервированном диапазоне 0..99. RegisterType со-

здает элемент в связанном списке записей TStreamRec.

См. также TStream.Get, TStreamPut, TStreamRec.

Переменная RepeatDelay — Drivers

Объявление RepeatDelay = 8;

Функция Определяет число квантов времени (1/18.2 часть секунды), которое должно быть известно перед генерацией событий evMouseAuto. Временной интервал между событиями evMouseAuto всегда составляет один квант.

См. также DoubleDelay, GetMouseEvent, константы evXXXX.

Переменная SaveCtrlBreak — Drivers

Объявление SaveCtrlBreak: Boolean = False;

Функция Процедура InitSysError сохраняет состояние Ctrl-Break DOS, проверяя эту переменную перед запрещением проверки Ctrl-Break DOS. DoneSysError восстанавливает Ctrl-Break DOS, проверяя значение, сохраненное в этой переменной.

См. также InitSysError, DoneSysError

Константы sbXXXX — Views

Функция Эти константы определяют различные области TScrollBar, в которых воспринимается отметка мышкой.

Функция TScrollBar.ScrollStep осуществляет преобразование этих констант в действительные значения шага скроллинга. Хотя она определена, константа sbIndicator никогда не передается в TScrollBar.ScrollStep.

Таблица 4.24.

Константы полосы скроллинга

Константа	Значение	Назначение
sbLeftArrow	0	Левая стрелка горизонтальной полосы скроллинга.
sbRightArrow	1	Правая стрелка горизонтальной полосы скроллинга
sbPageLeft	2	Левая страничная область горизонтальной полосы скроллинга
sbPageRight	3	Правая страничная область горизонтальной полосы скроллинга
sbUpArrow	4	Стрелка вверх вертикальной полосы скроллинга
sbDownArrow	5	Стрелка вниз вертикальной полосы скроллинга
sbPageUp	6	Верхняя страничная область вертикальной полосы скроллинга
sbPageDown	7	Нижняя страничная область вертикальной полосы скроллинга
sbIndicator	8	Индикатор на полосе скроллинга

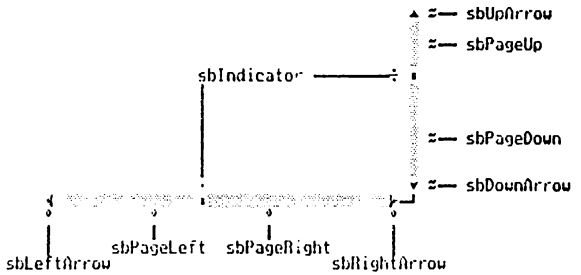


Рис. 4.5. Полоса скроллинга

Следующие значения могут быть переданы в функцию TWindow.StandardScrollBar:

Таблица 4.25.

Константы StandardScrollBar

Константа	Значение	Назначение
sbHorizontal	$0000	Полоса скроллинга горизонтальна
sbVertical	$0001	Полоса скроллинга вертикальна
sbHandleKeyboard	$0002	Полоса скроллинга реагирует на команды клавиатуры

См. также TScrollBar, TScrollBar.TScrollStep.

Переменная ScreenBuffer Drivers

Объявление ScreenBuffer: Pointer;
Функция Указатель на буфер экрана.
См. также InitVideo.

Переменная ScreenHeight Drivers

Объявление ScreenHeight: Byte;
Функция Устанавливается InitVideo и SetVideoMode в высоту экрана в строках для текущего экрана.
См. также InitVideo, SetVideoMode, ScreenWidth.

Переменная ScreenMode Drivers

Объявление ScreenMode: Word;
Функция Хранит текущий видеорежим. Изначально устанавливается инициализационным кодом модуля Drivers, ScreenMode может быть изменена использованием SetVideoMode. Значения ScreenMode обычно устанавливаются использованием мнемоник режима экрана smXXXX.
См. также InitVideo, SetVideoMode, smXXXX.

Переменная ScreenWidth Drivers

Объявление ScreenWidth: Byte;
Функция Устанавливается InitVideo в ширину экрана (число символов с строке).
См. также InitVideo.

Тип SelectMode Views

Объявление SelectMode = (NormalSelect, EnterSelect, LeaveSelect);
Функция Используется внутренне Turbo Vision.
См. также TGroup.ExecView, TGroup.SetCurrent.

Процедура SetVideoMode Drivers

Объявление procedure SetVideoMode(Mode: Word);
Функция Устанавливает видеорежим. Mode — одна из констант smCO80, smBW80 или smMono с необязательным smFont8x8 добавленным для выбора 43 или 50-строчного режима EGA или VGA. SetVideoMode инициализирует некоторые переменные как InitVideo (за исключением переменной StartupMode, на которую это

не воздействует). SetVideoMode обычно не вызывается напрямую. Вместо этого используйте TApplication.SetScreenMode, которая также устанавливает палитру программы.

См. также InitVideo, константы smXXXX, TApplication.SetScreenMode.

Константы sfXXXX — Views

Функция Эти константы используются для доступа к соответствующим битам полей TView.State. Поля TView.State никогда не должны изменяться напрямую; вместо этого вы должны использовать метод TView.SetState.

Значения Определены следующие флаги состояния:

Таблица 4.26.

Константы флагов состояния

Константа	Назначение
sfVisible	Установлен, если видимый элемент виден в своем владельце. Видимые элементы по умолчанию sfVisible. Методы TView.Show и TView.Hide могут использоваться для модификации sfVisible. При sfVisible видимый элемент не обязательно виден на экране, поскольку его владелец может быть невидим. Для проверки видимости на экране, проверьте бит sfExposed или вызовите функцию TView.Exposed.
sfCursorVis	Установлен, если курсор видимого элемента виден, в противном случае очищен. По умолчанию очищен. Методы TView.ShowCursor и TView.HideCursor могут использоваться для модификации sfCursorVis.

Константа	Назначение
sfCursorIns	Установлен, если курсор видимого элемента — сплошной блок, очищен, если курсор видимого элемента — линия. По умолчанию очищен. Методы TView.BlockCursor и TView.NormalCursor могут использоваться для модификации sfCursorIns.
sfShadow	Установлен, если видимый элемент имеет тень, в противном случае, очищен.
sfActive	Установлен, если видимый элемент это активное окно или подэлемент активного окна.
sfSelected	Установлен, если видимый элемент это текущий выбранный подэлемент внутри своего владельца. Каждый объект TGroup имеет поле Current, которое указывает на текущий выбранный подэлемент (или nil, если подэлементов не выбрано). Может быть только один выбранный подэлемент в TGroup.
sfFocused	Установлен, если видимый элемент сфокусированный. Видимый элемент — сфокусированный, если он выбран и все владельцы выше его также выбраны, т.е. если видимый элемент находится в цепи образованной указателями Current всех TGroup, начиная с TApplication (самый верхний видимый элемент в иерархии видимых элементов). Последний видимый элемент цепи — это конечное назначение для всех сфокусированных событий.
sfDragging	Установлен, если видимый элемент можно растягивать, в противном случае, очищен.

Константа	Назначение
sfDisabled	Установлен, если видимый элемент запрещен; очищен, если разрешен.
sfModal	Установлен, если видимый элемент — модальный. Всегда существует точно один элемент в выполняемой на Turbo Vision программе, обычно, объекты TApplication или TDialog. Когда видимый элемент начинает выполняться (через вызов ExecView), этот видимый элемент становится модальным. Модальный видимый элемент представляет вершину (корень) активного дерева событий, получая события и управляя ими до тех пор пока не вызван его метод EndModal. Во время этого «локального» цикла событий события передаются нижним подэлементам в дереве видимых подэлементов. События от этих нижних видимых элементов передаются по дереву, но не далее модального видимого элемента. См. также sfSelected, sfFocused, TView.SetState, TView.HandleEvent, TGroup.ExecView.
sfExposed	Установлен, если у видимого элемента прямой или косвенный владелец — объект TApplication и следовательно он может быть виден на экране. Метод TView.Exposed использует этот флаг при отсечении (клиппинге), определяя какая часть видимого элемента действительно видна на экране. См. также TView.Exposed.

Значения Биты флага состояния определены так:

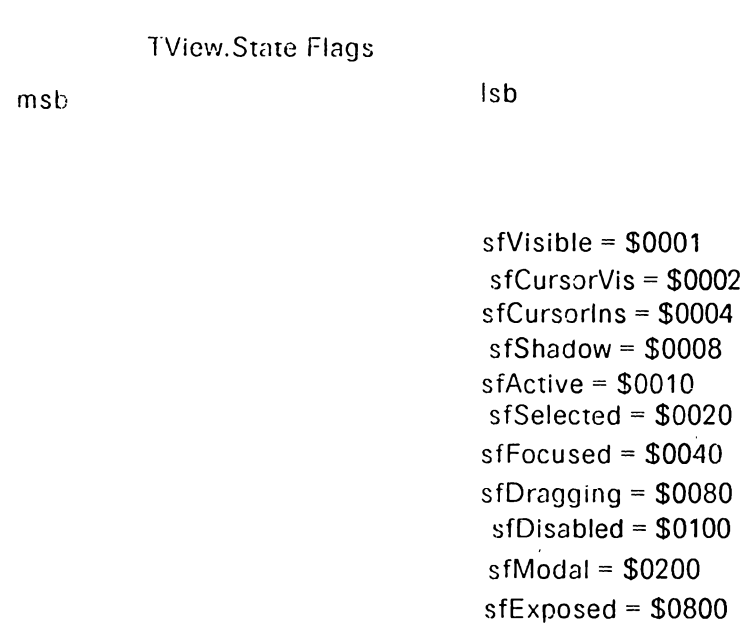


Рис. 4.6. Биты флага состояния

См. также TView.State.

Переменная ShadowAttr Views

Объявление ShadowAttr: Byte = $80;

Функция Эта переменная управляет цветом тени, доступной видимым элементам с установленным битом sfShadow. Тень — это обычно разреженная серая область, отображаемая прямо от краев видимого элемента с иллюзией трехмерности.

См. также ShadowSize.

Переменная ShadowSize Views

Объявление ShadowSize: TPoint = (X: 2; Y: 1);

Функция

Это значение управляет размером тени, которая доступна с установленным битом sfShadow. Тень это обычно разреженная серая область, отображаемая прямо от краев видимого элемента с иллюзией 3-D. По умолчанию, размер тени — 2 по оси X и 1 по Y. TProgram.InitScreen инициализирует ShadowSize: если режим экрана равен smMono, ShadowSize устанавливается в (0, 0). Иначе ShadowSize устанавливается в (2, 1), если только не установлен smFont8x8 (43- или 50-строчный режим), в этом случае устанавливается в (1, 1).

См. также TProgram.InitScreen, ShadowAttr.

Переменная ShowMarkers — Drivers

Объявление ShowMarkers: Boolean;

Функция Используется для указания, будут ли индикаторы размещаться вокруг активных элементов управления. TProgram.InitScreen устанавливает ShowMarkers в True, если монохромный видеорежим, иначе в False. Однако значение может быть установлено, при необходимости, в цветной или черно-белый режим.

См. также TProgram.InitScreen.
переменная SpecialChars.

Процедура ShowMouse — Drivers

Объявление procedure ShowMouse;

Функция ShowMouse уменьшает «счетчик невидимости» в драйвере мышки и делает курсор мышки видимым, если счетчик равен 0.

См. также InitEvents, DoneEvents, HideMouse.

Константы smXXXX — Drivers

Функция Эти мнемонические константы используются с SetVideoMode для установки соответствующего значения видеорежима в ScreenMode.

Значения В Turbo Vision определены следующие режимы экрана:

Таблица 4.27.

Константы режимов экрана

Константа	Значение	Назначение
smBW80	$0002	Черно-белый режим на цветном
smCO80	$0003	Цветной режим
smMono	$0007	Монохромный режим
smFont8x8	$0100	43- или 50-строчный режим

См. также SetVideoMode, ScreenMode.

Переменная SpecialChars Views

Объявление SpecialChars: array[0..5] of Char = (#175, #174, #26, #27, ' ', ' ');

Функция Определяет символы индикатора используемые для подсветки активного видимого элемента в монохромном видеорежиме. Эти символы отображаются, если переменная ShowMarkers равна True.

См. также переменную ShowMarkers.

Константы stXXXX Objects

Функция Существует два набора констант начинающихся с «st», которые используются потоками Turbo Vision.

Значения Следующие константы режима используются в TDosStream и TBufStream для определения режима доступа к файлу при открытии файла в потоках Turbo Vision:

Таблица 4.28.

Режимы доступа к потоку

Константа	Значение	Назначение
stCreate	$3C00	Создать новый файл
stOpenRead	$3D00	Открыть файл только на чтение
stOpenWrite	$3D01	Открыть файл только на запись
stOpen	$3D02	Открыть файл на чтение/запись

Следующие значения возвращаются TStream.Error в поле TStream.ErrorInfo, когда возникает ошибка потока:

Таблица 4.29.

Коды ошибок потока

Константа	Значение	Назначение
stOk	0	Нет ошибки
stError	−1	Ошибка доступа
stInitError	−2	Нельзя инициализировать поток
stReadError	−3	Чтение за концом потока
stWriteError	−4	Нельзя расширить поток
stGetError	−5	Get для незарегистрированного типа объекта
stPutError	−6	Put для незарегистрированного типа объекта

См. также TStream.

Переменная StartupMode Drivers

Объявление StartupMode: Word;

Функция Программа InitVideo сохраняет текущий режим экрана в этой переменной до переключения в режим экрана заданный в ScreenMode. DoneVideo восстанавливает режим экрана в значение, запомненное в StartupMode.

См. также InitVideo, DoneVideo, ScreenMode.

Переменная StatusLine App

Объявление StatusLine: PStatusLine = nil;

Функция Сохраняет указатель на строку статуса программы. Переменная StatusLine инициализируется в TProgram.InitStatusLine, вызываемой из TProgram.Init. Значение nil указывает, что в программе нет строки статуса.

См. также InitStatusLine.

Переменная StreamError Objects

Объявление StreamError: Pointer = nil;

Функция Если не nil, StreamError указывает на процедуру, которая вызывается методом Error потока при возникновении ошибки. Процедура должна быть дальней и использовать var параметр типа TStream, т.е. иметь объявление:

```
procedure MyStreamErrorProc(var S: TStream);
                                        far;
```

StreamError позволяет вам глобально перекрыть всю обработку ошибок потока. Чтобы изменить обработку ошибок для определенного типа потока, вы должны перекрыть метод Error этого потока.

Переменная SysColorAttr Drivers

Объявление SysColorAttr: Word = $4E4F;

Функция Цвет по умолчанию используется для вывода сообщений об ошибках обработчиком системных ошибок. На монохромных системах SysMonoAttr используется вместо SysColorAttr. Сообщения об ошибках с опцией

отменить/восстановить отображаются в строке статуса. Предыдущая строка статуса сохраняется и восстанавливается, когда условия разрешены.

См. также SystemError, SysMonoAttr.

Переменная SysErrActive Drivers

Объявление SysErrActive: Boolean = False;

Функция Указывает, активен ли обработчик системных ошибок в данный момент. Устанавливается в True через InitSysError.

Переменная SysErrorFunc Drivers

Объявление SysErrorFunc: TSysErrorFunc = SystemError;

Функция SysErrorFunc — это функция системной ошибки типа TSysErrorFunc. Функция системной ошибки вызывается при возникновении критической ошибки DOS или когда требуется смена диска на компьютере с одним гибким диском. ErrorCode — это значение от 0 до 15, как определено в таблице 4.30, а Drive — это номер устройства (0=A, 1=B и т.д.) для дисковых ошибок. По умолчанию функция системной ошибки — это SystemError. Вы можете установить свою функцию системной ошибки, назначая ее в SysErrorFunc. Функции системных ошибок не могут перекрываться.

Таблица 4.30

Коды функции системной ошибки

Код ошибки	Значение
0..12	Коды критических ошибок DOS
13	Плохой образ таблицы распределения файлов
14	Ошибка доступа к устройству

Код ошибки	Значение
15	Указание смены диска

Возвращаемые значения функции:

Таблица 4.31.

Значения, возвращаемые функцией системной ошибки

Возвращаемое значение	Назначение
0	Пользователь запросил повтор
1	Пользователь запросил отмену

См. также функцию SystemError, тип TSysErrorFunc, процедуру InitSysError.

Переменная SysMonoAttr — Drivers

Объявление SysMonoAttr: Word = $7070;

Функция Атрибут по умолчанию используется для вывода сообщений об ошибках обработчиком системных ошибок. На цветных системах SysColorAttr используется вместо SysMonoAttr. Сообщения об ошибках с опцией отменить/восстановить отображаются в строке статуса. Предыдущая строка статуса сохраняется и восстанавливается, когда условия разрешены.

См. также SystemError, SysColorAttr.

Функция SystemError — Drivers

Объявление function SystemError(ErrorCode: Integer; Drive: Byte): Integer;

Функция Функция системной ошибки по умолчанию. Она отображает одно из следующих сообщений об ошибке в строке статуса в зависимости от значения ErrorCode, используя атрибуты цвета, определяемые SysColorAttr или SysMonoAttr.

Таблица 4.32.

Сообщения функции SystemError

Код ошибки	Сообщение
0	Disk is write-protected in drive X Диск в дисководе X защищен от записи
1	Critical disk error on drive X Критическая ошибка диска в дисководе X
2	Disk is not ready in drive X Диск на дисководе X не готов
3	Critical disk error on drive X Критическая ошибка на дисководе X
4	Data integrity error on drive X Ошибка данных на дисководе X
5	Critical disk error on drive X Критическая ошибка на дисководе X
6	Seek error on drive X Ошибка позиционирования на дисководе X
7	Unknown media type in drive X Неизвестный тип носителя в дисководе X
8	Sector not found on drive X Не найден сектор на дисководе X
9	Printer out of paper Нет бумаги в принтере
10	Write fault on drive X Ошибка записи на дисководе X
11	Read fault on drive X Ошибка чтения на дисководе X
12	Hardware failure on drive X Аппаратная неисправность на дисководе X
13	Bad memory image of FAT detected В FAT обнаружен неверный образ памяти
14	Device access error Ошибка доступа к устройству

Код ошибки	Сообщение
15	Insert diskette in drive X Вставьте дискету в дисковод X

См. также SysColorAttr, SysMonAttr, SysErrorFunc.

Тип TByteArray Objects

Объявление TByteArray = array[0..32767] of Byte;
Функция Тип массива байт для общего использования при приведении типа.
См. также TStringListMaker.

Тип TCommandSet Views

Объявление TCommandSet = set of Byte;
Функция TCommandSet полезен для хранения произвольного множества, содержащего до 256 команд. Он позволяет выполнить простое тестирование, соответствует ли данная команда определенному критерию в программах обработки событий, и позволяет вам установить маски команд. Например, методы TView: EnableCommands, DisableCommands, GetCommands и SetCommands используют аргументы типа TCommandSet. Множество команд может быть объявлено и инициализировано с использованием синтаксиса Паскаля:

```
CurCommandSet: TCommandSet = [0..255] —
  [cmZoom, cmClose, cmResize, cmNext];
```

См. также cmXXXX, TView.DisableCommands, TView.EnableCommands, TView.GetCommands, TView.SetCommands.

Тип TDrawBuffer Views

Объявление TDrawBuffer = array[0..MaxViewWidth-1] of Word;
Функция Тип TDrawBuffer используется для объявления буферов для методов Draw различных ви-

димых элементов. Обычно данные и атрибуты сохраняются и форматируются строка за строкой в TDrawBuffer, а затем выводятся на экран:

```
var
  B: TDrawBuffer;
begin
  MoveChar(B, ' ', GetColor(1), Size.X);
  WriteLine(0, 0, Size.X, Size.Y, B);
end;
```

См. также TView.Draw, MoveBuf, MoveChar, MoveCStr, MoveStr.

Тип TEvent — Drivers

Объявление

```
TEvent = record
  What: Word;
  case Word of
    evNothing: ( );
    evMouse: (
      Buttons: Byte;
      Double: Boolean;
      Where: TPoint);
    evKeyDown: (
    case Integer of
      0: (KeyCode: Word);
      1: (CharCode: Byte);
    evMessage: (
    Command: Word;
    case Word of
      0: (InfoPtr: Pointer);
      1: (InfoLong: Longint);
      2: (InfoWord: Word);
      3: (InfoInt: Integer);
      4: (InfoByte: Byte);
      5: (InfoChar: Char);
end;
```

Функция Тип записи с вариантами TEvent играет фундаментальную роль в стратегии управления

событиями в Turbo Vision. Внешние события, такие, как события от клавиатуры и от мышки и события-команды, генерируемые взаимодействием видимых элементов, сохраняются и передаются как записи TEvent.

См. также evXXXX, HandleEvent методы, GetKeyEvent, GetMouseEvent

Тип TItemList — Objects

Объявление TItemList = array[0..MaxCollectionSize – 1] of Pointer;

Функция Массив общих указателей используется внутренне объектами TCollection.

Тип TMenu — Menus

Объявление
```
TMenu = record
  Items: PMenuItem;
  Default: PMenuItem);
end;
```

Функция Тип TMenu представляет один уровень дерева меню. Поле Items указывает на список TMenuItems и поле Default указывает на умалчиваемый элемент внутри этого списка (элемент выбирается по умолчанию при открытии этого меню). Объект TMenuView (потомками которого являются TMenuBar и TMenuBox) имеет поле Menu, которое указывает на TMenu. Записи TMenu создаются и удаляются с помощью подпрограмм NewMenu и DisposeMenu.

См. также TMebuView, TMenuItem, NewMenu, DisposeMenu, TMenuView.Menu поля

Тип TMenuItem — Menus

Объявление
```
TMenuItem = record
  Next: PMenuItem;
  Name: PString;
  Command: Word;
  Disabled: Boolean;
```

```
    KeyCode: Word;
    HelpCtx: Word;
    case Integer of
        0: (Param: PString);
        1: (SubMenu: PMenu);
  end;
end;
```

Функция

Тип TMenuItem представляет элемент меню, который может быть обычным элементом, подменю или строкой-разделителем. Next указывает на следующий TMenuItem в списке элементов меню или равен nil, если это последний элемент. Name указывает на строку, содержащую имя элемента меню, или равен nil, если элемент меню является строкой-разделителем. Command содержит событие-команду (см. константы cmXXXX) генерируемую при выборе элементов меню или 0, если элемент меню представляет подменю. Disable равно True, если элемент меню запрещен, False в противном случае. KeyCode содержит скэн-код горячей клавиши, связанной с элементом меню, или 0, если элемент меню не имеет горячей клавиши. HelpCtx содержит номер контекстной подсказки элемента меню (значение hcNoContext указывает, что элемент меню не имеет контекстной подсказки). Если элемент меню — это обычный элемент, Param содержит указатель на параметр строки (отображаемый справа от элемента в TMenuBox) или nil, если элемент не имеет параметра строки. Если элемент меню — это подменю, Submenu указывает на структуру подменю. Записи TMenuItem создаются использованием функций NewItem, NewLine и NewSubmenu.

См. также

TMenu, TMebuView, NewItem, NewLine, NewSubMenu

Тип TMenuStr — Menus

Объявление: `TMenuStr = string[31];`

Функция: Тип строки, используемый NewItem и NewSubMenu. Максимальный размер заголовка элемента меню — 31 символ.

См. также: NewItem, NewSubMenu

Тип TPalette — Views

Объявление: `TPalette = String;`

Функция: Тип строки, используемый для объявления палитр Turbo Vision.

См. также: GetPalette методы

Тип TScrollChars — Views

Объявление: `TScrollChars = array[0..4] of Char;`

Функция: Массив, представляющий символы, используемые для рисования TScrollBar.

См. также: TScrollBar

Тип TSItem — Dialogs

Объявление:
```
TSItem = record
   Value: PString;
   Next: PSItem;
end;
```

Функция: Тип записи TSItem обеспечивает односвязный список из PString. Подобные списки могут быть полезны во многих программах на Turbo Vision, где не требуется полная гибкость коллекций строк (см. TCluster, например). Функция NewSItem предназначена для добавления записей в список TSItem.

Тип TStatusDef — Menus

Объявление:
```
TStatusDef = record
Next: PStatusDef;
Min, Max: Word;
Items: PStatusItem;
end;
```

Функция Тип TStatusDef представляет определение строки статуса. Поле Next указывает на следующий TStatusDef в списке строк статуса или nil, если это последняя строка статуса. Min и Max определяют диапазон контекста подсказки, который соответствует строке статуса. Items указывает на список элементов строки статуса или nil, если в строке статуса нет элементов. Объект TStatusLine (строка статуса) имеет указатель на список записей TStatusDef и будет всегда отображать первую строку статуса, для которой текущая контекстная подсказка находится внутри диапазона Min и Max. Программа Turbo Vision автоматически корректирует строку статуса, вызывая TStatusLine.Update из TProgram.Idle. Записи TStatusDef создаются использованием функции NewStatusDef.

См. также TStatusLine, TProgram.Idle, функция NewStatusDef

Тип TStatusItem — Menus

Объявление
```
TStatusITEm = record
  Next: PStatusItem;
  Text: PString;
  KeyCode: Word;
  Command: Word;
end;
```

Функция Тип TStatusItem представляет элемент строки статуса, который может быть видим или невидим. Next указывает на следующий TStatusItem в списке элементов строки статуса или nil, если это последний элемент. Text указывает на строку, содержащую надпись элемента статуса (такую, как 'Alt-X Exit') или nil, если элемент статуса невидим (в таком случае элемент служит только для определения горячей клавиши). KeyCode содержит скэн-код горячей клавиши, связанной с

элементом статуса, или 0, если элемент статуса не имеет горячей клавиши. Command содержит команду-событие (см. константы cmXXX), генерируемую при выборе элемента статуса.

Функция записи TStatusItem не только в определении визуализации строки статуса, она также используется для определения горячей клавиши, чьи коды автоматически отображаются в команды. Метод TProgram. GetEvent вызывает TStatusLine.HandleEvent для всех событий evKeyDown. TStatusLine.HandleEvent сканирует текущую строку статуса на элементы, содержащие данный код клавиши и, если один из них найден, он преобразует это событие evKeyDown в событие evCommand со значением Command, данным в TStatusItem. Записи TStatusItem создаются использованием функции NewStatusKey.

См. также TStatusLine, NewStatusKey, TStatusLine.HandleEvent

Тип TStreamRec — Objects

Объявление
```
TStreamRec = ^TStreamRec;
  TStreamRec = record
  ObjType: Word;
  VmLink: Word;
  Load: Pointer;
  Store: Pointer;
  Next: Word;
end;
```

Функция Тип объекта Turbo Vision должен быть зарегистрирован TStreamRec до его загрузки или сохранения на объекте TStream. Подпрограмма RegisterTypes регистрирует тип объекта записью TStreamRec.

Поля в регистрационной записи потока определены так:

Таблица 4.33.

Поля записей потока

Поле	Содержимое
ObjType	Уникальный числовой идентификатор для типа объекта
VmtLine	Связь типа объекта с элементом VMT
Load	Указатель на конструктор Load объектного типа
Store	Указатель на метод Store объектного типа
Next	Указатель на следующую TStreamRec

Turbo Vision резервирует значения идентификаторов объектных типов от 0 до 999 для внутреннего использования. Программист может определить свои значения в диапазоне от 1000 до 65535. По соглашению, tStreamRec для типа объекта Txxxx называется Rxxxx. Например, TStreamRec для типа TCalculator называется RCalculator, как показано в следующем коде:

```
type
TCalculator = object(TDialog)
constructor Load(var S: TStream);
procedure Store(var S: TStream);
...
end;

const
RCalculator: TStreamRec = (
ObjType: 2099;
VmtLink: Ofs(TypeOf(TCalculator)^);
Load: @TCalculator.Load;
Store: @TCalculator.Store);

begin
RegisterType(RCalculator);
```

```
...
end;
```

См. также RegisterType

Тип TStrIndex — Objects

Объявление `TStrIndex = array[0..9999] of TStrIndexRec;`
Функция Используется внутренне TStringList и TStrListmaker.

Тип TStrIndexRec — Objects

Объявление
```
TStrIndexRec = record
  Key, Count, Offset: Word;
end;
```
Функция Используется внутренне TStringList и TStrListmaker.

Тип TSysErrorFunc — Drivers

Объявление `TSysErrorFunc = function(ErrorCode: Integer; Drive: Byte): Integer;`
Функция Определяет тип и функцию обработчика системной ошибки.
См. также SysErrorFunc, SystemError, InitSysError, DoneSysError

Тип TTerminalBuffer — TextView

Объявление `TTerminalBuffer = array[0..65519] of Char;`
Функция Используется внутренне TTerminal.
См. также TTerminal

Тип TTitleStr — Views

Объявление `TTitleStr = string[80];`
Функция Используется для объявления строк текста для заголовков окон.
См. также TWindow.Title

Тип TVideoBuf — Views

Объявление `TVideoBuf = array[0..3999] of Word;`

Функция Используется для объявления видеобуферов.
См. также TGroup.Buffer

Тип TWordArray — Objects

Объявление TWordArray = array [0..16383] of Word;
Функция Тип массива слов для общего использования.

Константы wfXXXX — Views

Функция Эти мнемоники определяют биты в поле Flags объектов TWindow. Если биты установлены, окно будет иметь соответствующие атрибуты: окно может перемещаться, изменять размер, закрываться или масштабироваться.
Значения Флаги окна определены так:

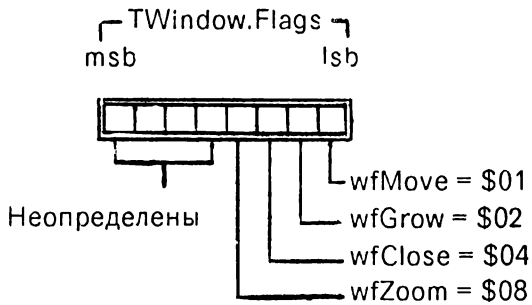


Таблица 4.34.

Константы флагов окна

Константа	Значение	Назначение
wfMove	$01	Окно можно перемещать

Константа	Значение	Назначение
wfGrow	$02	Можно изменять размеры окна соответствующей кнопкой в левом правом углу
wfClose	$04	Рамка окна имеет закрывающую кнопку
wfZoom	$08	Рамка окна имеет кнопку масштабирования

Если отдельный бит установлен, соответствующая возможность подключается, в противном случае, эта возможность запрещена.

См. также TWindows.Flags

Константа wnNoNumber — Views

Объявление wnNoNumber = 0;

Функция Если поле TWindow.Number содержит эту константу, это указывает, что окно не может иметь номера и не может быть выбрано через Alt-номер. Если поле Number между 1 и 9, номер окна отображается, и выбор по Alt-номер доступен.

См. также TWindow.Number

Тип WordRec — Objects

Объявление WordRec = record Lo, Hi: Byte; end;

Функция Утилитная запись, позволяющая доступ к младшему и старшему биту слова.

См. также LongRec

Константы wpXXXX — Views

Функция Эти константы определяют три стандартных цвета, отражая их назначение. По умолчанию, объект TWindow имеет палитру wpBlueWindow. По умолчанию, объект TDialog имеет палитру wpGrayWindow.

Значения Три стандартных палитры окна определены:

Таблица 4.35.

Стандартные палитры окна

Константа	Значение	Назначение
wpBlueWindow	0	Желтый на голубом
wpCuanWindow	1	Синий на бирюзовом
wpGrayWindow	2	Черный на сером

См. также TWindow.Palette, TWindow.GetPalette

Научно-проектная фирма «ДИАЛЕКТИКА» предлагает книги по новейшим технологиям программирования, которые будут изданы в IV квартале этого года:

Книга Питера Нортона и Поля Йао
« Программирование на BORLAND C++ для WINDOWS »
536 стр., твердый переплет

Аннотация

Одной из сложнейших задач, появляющихся при разработке прикладных программ для WINDOWS, является упорядочение громадного числа обращений к интерфейсу прикладных программ среды WINDOWS (API). Эта задача значительно упрощается при использовании разработанной фирмой BORLAND библиотеки ObjectWindows (OWL), однако для создания «послушных» прикладных программ, полностью соответствующих стандартам WINDOWS, необходимо знать как эти программы сочетаются, и как происходит вызов различных функций WINDOWS.

Книга предназначена для программистов, чем объясняется большое количество листингов программ. Однако эта книга не просто набор листингов программ, так как в ней таакже подробно рассматриваются концепции и философия WINDOWS.

Borland C++ дает Вам средства управления WINDOWS ; книга же дает Вам навыки эффективного использования этих средств. Рассматривая библиотеку ObjectWindows, авторы наглядно демонстрируют, как быстро и эффективно можно создавать прикладные программы с использованием всех предоставляемых WINDOWS средств — текста, графики, меню, диалоговых окон и других.

Несмотря на то, что большинство прикладных программ для WINDOWS сейчас пишутся на C, можно с уверенностью предсказать, что C++ скоро займет доминирующее положение. C++ дает возможность добавления объектного программирования к уже написаным на C системам. Именно по этой причине была создана версия книги для C++.

Эта книга написана для программистов, работающих с языком C++, с целью обучения программированию в среде

WINDOWS. В книге приводятся тексты программ, дающие возможность более полного ознакомления с используемой системой и ее возможностями.

Книга состоит из шести частей.

- В первой части приводится краткая история WINDOWS, а также описываются три новых дополнения, с которыми сталкивается программист: создание программ, управляемых сообщениями, создание и управление графическим выводом, использование объектов пользовательского интерфейса.
- Во второй части рассматриваются проблемы, связанные с созданием минимальной для WINDOWS программы, а так же некоторые базовые положения программирования в среде WINDOWS . В этой части также рассматривается структура, общая для всех программ в WINDOWS.
- В третьей части описывается интерфейс графических устройств (GDI). Этот интерфейс используется для создания аппаратно-независимого графического вывода.
- В четвертой части рассматриваются три ключевых объекта пользовательского интерфейса: меню, окна, диалоговые окна. В этой части описываются внутренние свойства каждого из объектов и основные способы их использования.
- Пятая часть посвящена средствам ввода. В этой части приводится подробное описание процесса ввода данных с физического устройства через системные буфера в программу.
- В шестой части рассматриваются особенности операционной системы. Эта часть включает два больших раздела: линкование в памяти и динамическое линкование. Одной из отличительной особенности версии WINDOWS 3.0 являются изменения при использовании памяти. Для того, чтобы понять понять эти изменения, приводится подробное описание каждого из операционных режимов WINDOWS.

Эта книга станет важной частью Ваших знаний о программировании в среде WINDOWS.

Справочное руководство по FoxPro 2.0 в трех томах, по 520 стр., твердый переплет

Аннотация

Данное руководство предлагает читателю полное описание одной из самых лучших систем управления базами данных для персональных компьютеров FoxPro 2.0 . Трехтомное издание хорошо иллюстрировано, содержит много программ, написанных в системе FoxPro, богато примерами, показывающими ее лучшие качества и возможности.

Предназначено для широкого круга специалистов по вычислительной технике и базам данных.

Книга Гради Буча «Объектно-ориентированное проектирование с примерами применения», 536 стр., твердый переплет

Аннотация

Предлагаемая Вашему вниманию книга "Object Orinted Design with Application" by Grady Booch" (в издании на русском языке — «Объектно-ориентированное проектирование с примерами применения») вышла в издательстве "The Benjamin/Cummings Publishing, Inc." (США) в 1991 году и является одной из самых продаваемых.

Об авторе. Гради Буч — основатель и директор американской фирмы Object-Orinted Products at Rational, образованной в 1980 году. Был одним из первых в США, кто начал использование объектно-ориентированного метода проектирования с применением различных объектных и объектно-ориентированных языков программирования. Г-н Буч является членом редколлегий американских журналов: Object Magazin, Journal of Object-Orinted Programming, HotLine of Object-Orinted Technology.

О книге. «Объектно-ориентированное проектирование с примерами применения» представляет собой первое полное изложение объектно-ориентированной методологии и ее компонент. Книга разделена автором на три основные части.

Часть первая — ***Концепции.*** В этой части обсуждаются свойства сложных систем, рассматривается понятие декомпозиции, какую роль при этом играют абстракции, иерархия. Понятие конструирования системы определено как процесс создания модели реального объекта. Автором рассмотрены фундаментальные основы объектного подхода и произведен анализ развития языков программирования. В качестве элементов объектного подхода выделяются процессы объектно-ориентированных анализа (OOA), проектирования (OOD) и программирования (OOP). Приведены направления где практические результаты использования этого подхода уже получены. Автор подчеркивает значения

Классов и Объектов в процессе проектирования и дает рекомендации по оценке качества этих абстракций. Содержатся примеры структур классов и объектов, оптимизирующих программные системы. Отдельная глава посвящена выработке правил классификации классов и объектов, рассматриваются три основных типа классификаций: Классическое распределение по категориям; Концептуальное группирование; Теория прототипирования.

Часть вторая — ***Методология.*** Здесь рассматривается система обозначений, позволяющая описать проект не прибегая к средствам языка реализации проекта. Автором описан собственно процесс объектно-ориентированного проектирования. выделены основные этапы этого процесса, рассмотрены содержание и разультаты каждого из них. В этой части также рассмотрены практические аспекты конструирования реальных программных систем, а также основные инструменты, облегчающие труд программиста. Даны некоторые советы, как перейти к объектно-ориентированному проектированию.

Часть третья — ***Примеры применения.*** В этой части приводятся примеры применения на следущих языках программирования: Smalltalk, Object Pascal, C++, Common Lisp Object System, Ada.

Особое достоинство книги — обширная библиография на сорока страницах, в которой представлены публикации по всем аспектам технологии объектно-ориентированного программирования.

Книга расчитана на профессиональных программистов, руководителей больших программных проектов и студентов, будущая профессиональная деятельность которых связана с разработкой сложных программных систем.

Книга Михеля Шумана «Turbo Pascal для Windows» 396 стр., твердый переплет

Анотация

Поздравляем Вас с решением заниматься программированием под Windows. Так как этот продукт в 1990–92 гг. сделал много шума ,то можно считать, что через несколько лет почти каждая программа будет работать под Windows.

Программирование под Windows было еще недавно делом очень дорогим, т.к. были необходимы дорогостоящие С-компиляторы и кроме того специальный пакет для создания программ под Windows.С другой стороны создание программного обеспечения под Windows было очень трудоемким процессом, т.к. путь от идеи к программе занимал слишком много времени.

Почти 10 лет назад фирма Borland выбросила на рынок свой легендарный компилятор Turbo Pascal. В то время память компьютеров считалась еще в килобайтах и тактовая частота не превышала 10 MHz.Уже тогда Turbo Pascal обгонял свое время,обладая необычайным комфортом и очень высокой скоростью компиляции, поэтому название Turbo Pascal происходит от слова Turbo, что означает быстрый.

Прошло время, но Turbo Pascal остался современным и даже более того, вырвался вперед опережая конкурентов.

Среди компиляторов Pascal для PC Turbo Pascal находится на первом месте и является некоторым стандартом, на который ориентировалась даже фирма Microsoft при развитии своего компилятора QuickPascal.

Что Вас ждет в этой книге ?

В первом разделе мы рассматриваем Windows и функции, которые она Вам предоставляет.Объясняются стандарты,которых должна придерживаться каждая программа под Windows и все основные функции и понятия для среды Windows.

Во втором разделе рассматривается пакет Turbo Pascal для Windows,его инсталяция,конфигурация, а также дается краткое руководство пользователя. Здесь же мы будем

создавать нашу первую программу, компилировать и испытывать ее в работе.

В третьем разделе Вы найдете основы языка программирования Turbo Pascal, что наверное очень интересно для новичков, а также для тех, кто переходит с других языков программирования.

Объектно-ориентированное программирование (ООР) является тематикой четвертого раздела,который постепенно ознакомит Вас с идеями и методами этой новой технологии программирования.

В пятом разделе мы создадим проект реально работающего приложения для Windows. Здесь же мы используем редактор ресурсов Whitewater Resource Tool.

В шестом разделе мы остановимся на особенностях программирования под Windows и используем предлагаемые Windows объекты и функции.

Show me, don't tell me! (Что означает «Лучше один раз увидеть, чем сто раз услышать!»).Следуя этой поговорке мы создаем в седьмом разделе полное Windows-приложение.

В этой книге Вы найдете все этапы создания реального проекта: от постановки задачи до программирования ее.

Работа со встроенным Turbo Debugger for Windows (TDW) для поиска и обнаружения сложных ошибок является темой восьмого раздела.

Полный справочник Turbo Pascal for Windows и важные функции самой системы Windows Вы найдете в девятом разделе. Кроме того книга имеет глоссарий в котором Вы можете найти все термины, используемые в этой книге.

Книга снабжена дискетой с примерами программ, рассматриваемых в этом издании.

Принимаются заявки по адресу : Украина, г.Киев ул.Пархоменко, 14 салон КОМТЕХ с грифом «Библиотека программиста».

И не забудьте вложить почтовую открытку с указанием названия интересующей Вас книги, количества экземпляров и обратным адресом.

Оглавление

Глава 3. Справочник по объектам

Учебное издание

Turbo Vision для языка Pascal.

Справочник

Подписано к печати 27.10.92. Формат 84×108/32 Бумага офсетная. Гарнитура таймс. Печать офсетная. Усл. печ.-листов 15,12 . Уч.-изд. листов 16,2.

Цена договорная. Зак. 0212156.

Отпечатано с компьютерного набора на комбинате печати издательства «Пресса Украины»
252047, Киев-47, проспект Победы, 50.

НАУЧНО-ПРОЕКТНАЯ ИССЛЕДОВАТЕЛЬСКАЯ ФИРМА „ДИАЛЕКТИКА“

- РАЗРАБОТКА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ПРОГРАММИСТА;
- ПРОВЕДЕНИЕ ИЗЫСКАНИЙ В ОБЛАСТИ КОГНИТОЛОГИИ И ИССКУСТВЕННОГО ИНТЕЛЛЕКТА;
- ФОРМАЛИЗАЦИЯ И РЕАЛИЗАЦИЯ НОВЫХ ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ;
- ВЫПУСК ЛИТЕРАТУРЫ СООТВЕТСТВУЮЩЕЙ ТЕМАТИКИ;
- ОКАЗАНИЕ ПОМОЩИ В РАЗРАБОТКЕ И ВНЕДРЕНИИ НОВЫХ ИДЕЙ;
- ОРГАНИЗАЦИЯ МЕРОПРИЯТИЙ ПО ОБЪЕДИНЕНИЮ УСИЛИЙ ПРОГРАММИСТОВ В РЕШЕНИИ ПРОБЛЕМ СПЕЦИФИЧНЫХ ДЛЯ ДАННОЙ ОБЛАСТИ;